Learning PUFs from Unsigned Stability: A Warning on Tolerable Leakage

Niklas Stein[0009−0006−0278−6651] and Michael Pehl[0000−0001−6100−7714]

Technical University of Munich, School of Computation, Information and Technology, Chair of Security in Information Technology, 80333 Munich, Germany
{niklas.stein, m.pehl}@tum.de

Abstract. Physical Unclonable Functions (PUFs) are an upcoming security primitive for silicon chips, offering high attack resilience at low production cost, especially for resource-constrained devices. Common implementations of these PUFs utilize challenge-response behavior, promising a significant number of bits per area. However, this family of PUFs is also known for its vulnerability to machine-learning attacks. While training based on challenge-response pairs is not possible in several scenarios like in key-storage applications, we demonstrate in this work another pitfall: Based on the example of Arbiter and Loop PUF, we demonstrate that the leakage of stability information from a side-channel attack can be exploited to successfully break the PUF, although the actual secret is not visible to an attacker. Results from numerical PUF models and actual measurements confirm the applicability of the attack. The method can also be combined with previously published attacks exploiting helper data leakage for machine learning. The attack highlights the importance of limiting the challenges applied to a PUF that leaks stability information to Hadamard challenges, which is an effective countermeasure not only against this novel but also against previous machine-learning attacks exploiting other forms of leakage.

Keywords: Physical Unclonable Function · Side-Channel Analysis · Loop PUF · Arbiter PUF · Fuzzy Extractor · Machine Learning Attack

1 Introduction

The rise of IoT devices brings a series of challenges to both chip designers and security engineers. Since these devices are handed out to third parties in large quantities, they will end up in the hands of skilled attackers or competitors and have to face almost any kind of physical attack. On the other hand, many devices like RFID tags or smart home accessories are short-lived and disposable products. This puts a tight constraint on the production cost and available semiconductor technologies. Individual random keys per device are required to mitigate the damage of a successful break, implying the need for secure key storage in such devices. Where classical methods for such secure key storage are unsatisfactory, too expensive, or not available, Physical Unclonable Functions

(PUFs) provide an alternative: They measure random manufacturing variations of silicon devices and map them to bit values usable for challenge-response authentication or – after some error correction – as a secret key.

One important category of silicon PUFs is the class of delay-based PUFs. These PUFs measure the process-variation-dependent time it takes for a signal to travel through the circuit under test. Since the expected value of the delay depends highly on the environmental conditions, it is common practice to measure in a differential way so that the expected difference has zero mean. Examples for such an implementation of a differential measurement for PUFs include the pair-wise comparison of ring oscillators (ROs) in case of RO PUFs [23], the comparison of two delay paths in case of Arbiter PUFs [6], or the comparison of the circuit under challenge and inverse challenge in case of the Loop PUF [4].

The measured analog (or, under finite measurement resolution, quasi-analog) value received from a delay-based PUF is quantized into a binary value for further use. The most straightforward quantization scheme for differentially measured delay paths is based on the signum function: One bit is derived from the sign of the difference between two path delays or frequencies, indicating which of the two observed paths was faster. For PUFs using sign-based quantization, Side-Channel attacks have been presented that reveal the secret bit for unprotected designs, e.g., for RO PUFs [14], Loop PUFs [27], TERO PUFs [26], or Arbiter PUFs [1]. Also, corresponding countermeasures were proposed that hide the output bit from the attacker [24]. Nevertheless, in particular, for oscillator-based PUFs, SCA still reveals the absolute value of the amplitude. This might be seen as an uncritical form of leakage when the sign-bit is hidden from the attacker since the absolute value of the amplitude is not directly related to the secret sign-bit of a difference. However, our case study of the Loop PUF in this work reveals a potential remaining issue.

In addition to sign-based quantization, more sophisticated schemes exist to quantize into a higher-order alphabet or to quantize in a way that makes the response more reliable. In this case, the expected distribution of differential analog values is split into several equiprobable intervals. For schemes like [3, 7], these intervals are interpreted as multi-bit words, allowing for the generation of more secret output bits with fewer quasi-analog PUF responses. Assigning single bits in a specific way to multiple intervals and storing some helper data for decoding results in the Two-Metric Helper Data scheme [5], which can significantly reduce errors in the output.

However, the mentioned more advanced quantization methods are difficult to protect against Side-Channel Analysis (SCA) attacks: while the sign of a delay difference can be protected quite well with the countermeasures discussed later, the absolute magnitude turns out to be easy to obtain in several channels and inherently hard to protect. Under this leakage of the magnitude – and considering in some cases the Helper Data – only one or two intervals, and, thus, codewords are possible, reducing the entropy down to 0 to 1 bits per word [25]. Research targeting the absolute magnitude of a difference in a higher-order or Two-Metric Helper Data setting already exists, while this kind of leakage remains widely

unconsidered for sign-based quantization. In addition, sign-based quantization might be, in several cases, the preferred option due to area efficiency [16]. Thus, this work focuses on sign-based quantization.

Contribution: This work combines SCA and machine-learning (ML) attacks on PUFs and shows how to exploit side-channel leakage to construct a precise model of the PUF. Specifically, modeling is feasible although the actual PUF response is hidden from the attacker and only the absolute magnitude of the measured analog PUF response is revealed in the side channel.

In the course of achieving this main contribution,

- We discuss on a theoretical level how side-channel leakage is used to train a machine-learning model, introducing a novel two-stage likelihood-based learning approach.
- We show the practical applicability of the attack for a Loop PUF implemented on an FPGA and measured via a power side channel, providing the first attack on the temporal masking scheme
- We discuss the application to Arbiter PUFs on a simulation level.
- We introduce a combined attack strategy for key storage devices with available helper data.
- We discuss potential countermeasures and how they are effective against such an attack.

Structure: The rest of this work is structured as follows: Section 2 discusses the leakage source, exploited by the SCA, highlighting in particular our example for the case study: the Loop PUF. Section 3 introduces the linear model for delay-based PUFs that we use in Section 4 to introduce the theory behind the attack strategy and the modeling concept of the PUF itself. Afterwards, Section 5 shows how the strategy is applied in practice and provides corresponding results. Countermeasures to our findings are discussed in Section 6 before we conclude in Section 7.

2 Stability Leakage in Hardened PUF designs

Quantization is usually achieved for delay-based PUFs by comparing an analog measurement against a decision bound (0 for sign-based quantization). In this, analog values have different distances to the decision boundary. Almost equal delays, e.g., result in a differential value close to zero, and, thus, a small amount of inevitable measurement noise suffices to flip the sign, resulting in an erroneous output for sign-based quantization. In contrast, large differences are less susceptible to noise effects since measurement noise with high magnitude is unlikely to occur. This known effect has been used, e.g., for soft-decision decoding in the context of PUF-based key storage [9].

The probability of a PUF's binary output value (the PUF response) reproducing its expected mean value is considered the reliability of a PUF. Attacks based on reliability are widely documented in Challenge-Response Pair (CRP)

settings [2, 15]. But similar to machine-learning attacks based on pure CRP behavior [19], the calculation of reliability usually relies on observing the responses. Thus, such attacks are costly if precise reliability information is needed, and not applicable if the responses are hidden from an attacker. Such cases of a hidden response occur, e.g., for key storage scenarios or for the inner layer of an Interpose PUF [15].

In this work, we will assume such an attacker with physical access to the device, but without the availability of CRPs and with basic protection mechanisms already in place. For this, we do not rely on the observation of the response value of a PUF but use side-channel information instead: We focus on the leakage of the distance between the (differential) quasi-analog response value and the decision boundary without knowing the secret sign. The next subsection discusses that this leakage is inherently harder to prevent using the example of a Loop PUF and an Arbiter PUF.

2.1 Temporal masking Loop PUF

The Loop PUF [4] as outlined in Fig. 1 is an easy-to-design challenge-dependent ring oscillator. It is constructed from a single RO and a counter to measure the oscillation frequency by counting periods within a fixed time. The ring has n delay elements, an inverting feedback path, and a trigger to switch the oscillation on or off (inversion and trigger are in the figure merged into a NAND-gate). Each delay element can be configured by a bit of the challenge C to select between two different paths, realized in Fig. 1 by 2-to-1 multiplexers with both inputs connected to each other. The configuration of the loop does not change the logical behavior but decides which transistors are passed by the oscillating edge and, thus, influences the frequency of the ring. The differential measurement is obtained from the same oscillator by applying not only C but also the bit-inverse challenge $\neg C$ in separate measurement runs and comparing the two frequencies. For this purpose, the counter can be configured to count downwards from the previous measurement in two's complement to eliminate additional subtraction logic.

The number and choice of the challenges can influence the quality of the output. As described in [18], only the first n orthogonal challenges actually contain one bit of entropy per output bit. After this, only pairwise orthogonal additions to this set can be found, which contain slightly less entropy. Following this idea over large sets, the total entropy would be expected to converge to $2n$ bits.

A side-channel attack on the unprotected Loop PUF was shown in [27]: The oscillation frequency of the ring is emitted by power and EM side channels. By applying a Short-Time Fourier Transform (STFT), the exact frequency of the Loop PUF under each challenge is obtained by observing each time slot where a different challenge is applied separately. The power spectrum can be averaged over many traces since the challenge sequence is identical on every readout, and thus, a high signal-to-noise ratio can be obtained even under field conditions. When the mapping of the time slots for the different challenges to the pairs of C

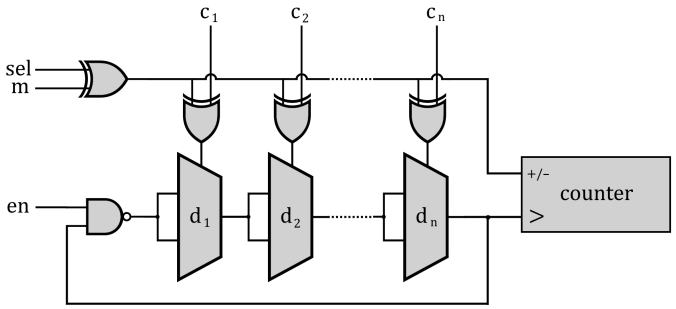

Fig. 1: Architecture of the masked Loop PUF. The delay elements of the oscillator d_1 to d_n are configured by the challenge bits c_1 to c_n. The mask bits m decide whether the normal or inverse measurement is executed first, the other measurement is then selected with *sel*.

and $\neg C$ is known, the secret bits can be guessed with high accuracy by relative comparison of the leaked frequencies.

A practical countermeasure to prevent leakage is to shuffle the evaluation order of the challenges. However, a true shuffling of all challenges causes an enormous hardware overhead compared to the small size of the PUF [25]. It would require the implementation of a non-repetitive queue like Fisher-Yates Shuffle, which consumes at least $\lceil log_2(n!) \rceil$ fresh random bits from a True Random Number Generator (TRNG). Furthermore, to revert the output bits back into the expected constant order, a form of random-access storage would be needed.

As a more lightweight alternative, the temporal masking scheme was proposed [24]. For this protection method, only the measurement order within each pair of challenge and complementary challenge (C, $\neg C$) is exchanged, while the order between the pairs remains constant. This method only requires minimal hardware overhead: One mask-bit per challenge to decide if C or $\neg C$ is used first for a specific pair. Overall, only n bits of randomness are needed for this method, which are obtained from either an external TRNG or by harvesting phase jitter noise from the Loop PUF oscillator. In the latter case, an unprotected dummy round is needed for the initialization, for which the output is discarded, and all subsequent rounds are protected through the randomness harvested when generating the respective previous PUF bit.

Unlike true shuffling, temporal masking still leaks the distance between the Loop PUF's frequency under C and $\neg C$ in the side channel. This is a deliberate design choice for sign-based quantization since the implementation is much more area-efficient. As shown in Figure 2, this leakage of the frequency difference can be extracted with high accuracy. For this experiment, a Loop PUF with 32 stages was implemented on a Xilinx Artix 7 FPGA and 200 power traces were collected. The power spectrum of each trace over time is calculated by a STFT. The resulting waterfall plots are then averaged over all traces to remove all zero-mean measurement noise. This reveals the frequency of the oscillator at each

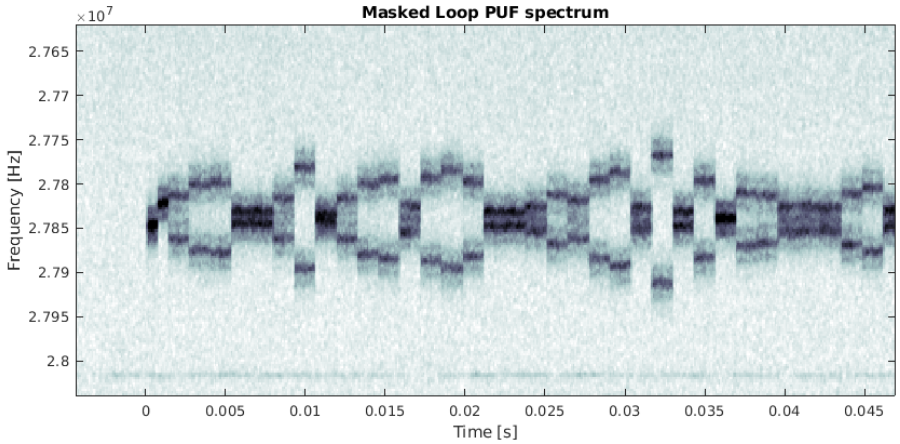

Fig. 2: Power spectrum of the masked Loop PUF. Each time slot of 1.3 ms aside the dummy round shows the superimposed frequency of C and $\neg C$.

time slot as a dark bar of high power. Since the mask is different in each trace, the time slots in each trace would randomly contain either C or $\neg C$. Averaging over all traces, both frequencies (for C or $\neg C$) are superimposed into two time slots. Thus, the plot shows: the sign of the difference cannot be extracted, but the absolute difference between frequencies is visible. Please note that – as this implementation harvests randomness from the phase jitter of the PUF itself – the first dummy challenge is unmasked, similar to every challenge on an unprotected Loop PUF.

2.2 Arbiter PUFs

Another common implementation of delay-based PUFs is the Arbiter PUF [6]. Their design aims to evaluate only a single edge on the delay line rather than oscillations. To obtain differential measurements, two similar delay lines are needed. Common implementations use crossover wiring as sketched in Fig. 3. As the delay difference is typically in the range of a few picoseconds, a specialized analog circuit is needed for the quantization, hence the name Arbiter. Since this circuit is also subjected to the same manufacturing tolerances, only a few methods like metastable SR latches are suitable and only sign-based quantization is practical. As these designs are generally more prone to noise, they are not a primary choice for key storage designs. However, some variants using hidden intermediate values like feed-forward and interpose PUFs are of interest for strong responses.

Where machine learning of CRPs is not possible, pure power side-channel attacks on Arbiter PUFs are mostly restricted to simple power analysis (SPA) of the output registers [1, 10]. This leakage occurs when the secret is converted from the complementary outputs of the latch to single rail logic for storage in e.g. a D flip-flop. As lightweight countermeasures, methods from dual-rail

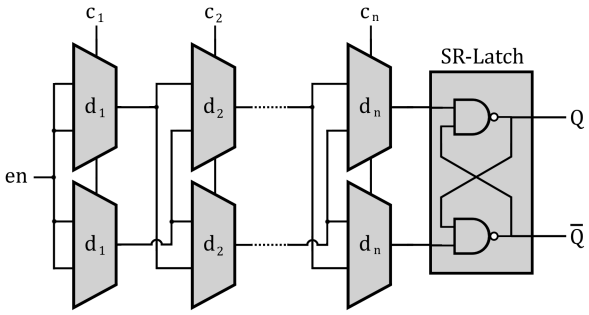

Fig. 3: Typical architecture of Arbiter PUFs. A single edge is split in two paths, each subjected to delay elements d_1 to d_n. An arbitration gate will detect which edge arrives first.

implementations have been proposed. By equalizing the capacitive load on Q and $\overline{Q}$ and by random pre-charging of registers, this leakage of the signed output can be significantly reduced [11].

However, our simulative experiments show that even ideal dual-rail Arbiter still leak the unsigned stability in the power side channel. This attack focuses on the arbitration latch itself, not the subsequent data paths. For this, a 64-stage device like Fig. 3 was created with the IHP 130nm Open PDK using the standard cells `sg13g2_mux2` and `sg13g2_nand2`. The parasitics are symmetric as in an ideal placement and both latch outputs have equal load; An attack by SPA on this subcircuit would not be possible. The analog NGSPICE simulation is capable of representing PUF behavior when Monte-Carlo transistor mismatch is enabled.

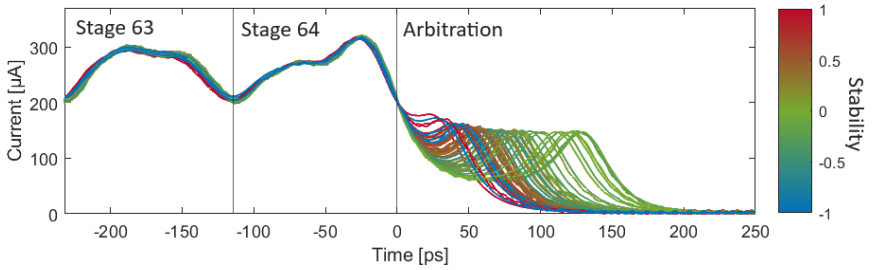

Fig. 4: Superimposed power traces of the Arbiter PUF. It can be noted that the time and power consumption depends on the absolute stability.

A section of these simulated traces is shown in Figure 4. It can be observed that the power profile of the arbitration latch changes with the (unsigned) stability. When the signal edges traversing the delay chain for one challenge arrive at the upper and, respectively, lower arbiter inputs with a large delay difference, the latch leaves its metastable state quickly. However, when the edges arrive

at the same time, the latch remains in its metastable state until small current differences finally sum up, resulting in a higher crossbar current and a later peak.

This correlation is nonlinear and – unlike the attack on the Loop PUF – would require some profiling to be practical. It is expected to require a large number of measurements, similar to the SPA attacks, to achieve a sufficient SNR. But since this leakage is part of the underlying mechanics of a PUF, the only viable protection mechanism would be an expensive shuffling of the execution order.

3 Linear Model of the Loop PUFs

The underlying structure and behavior of simple delay-based PUFs is highly linear, and as such they can be approximated well by linear models. Such linear models were originally described by [12] for Arbiter PUFs and by [28] for Bistable and Twisted-Bistable Ring PUFs. Here we transfer the idea to the Loop PUFs and describe a linear model for this PUF type we use as an example.

The delay of each Loop PUF stage is expressed as a delay of either the upper or lower path, where the decision of which path is made depending on the corresponding challenge bit.

$$d_x(c_x) = \begin{cases} u_x, & \text{if } c_x = 1 \\ v_x, & \text{if } c_x = 0 \end{cases}$$

The overall delay and frequency of the ring can then be expressed as the sum over all elements.

$$d(C) = \sum_{x=1}^{n} (d_x(c_x)) \quad \text{and} \quad f(C) = \frac{1}{d(C)}$$

The secret analog value s depends on the differential pair of normal and bit-inverse challenge.

$$s(C) = d(C) - d(\neg C) = \sum_{x=1}^{n} (d_x(c_x)) - \sum_{x=1}^{n} (d_x(\neg c_x))$$

As both paths of each delay element are used in either the normal or inverse measurement, and the rest of the path (e.g., interconnects between stages and feedback) remains constant, the secret analog value s can be expressed as a dot product of a challenge C in {-1,1} and the delay difference per element δd_x.

$$\begin{aligned} C \in \{1, -1\} &= C - \neg C \\ \delta d_x &= u_x - v_x \\ s &= C \cdot \delta d \end{aligned} \tag{1}$$

This results in a linear model with n variables for an n-stage Loop PUF. The secret bit under sign-based quantization corresponds to $sign(s)$, and the

unsigned stability is proportional to $abs(s)$. A more precise model could consider that the counter equals $1/d(C) - 1/d(\neg C)$, but the numerical difference is negligible in this setting.

Following the idea of formulating the analog secret as a dot product $C \cdot \delta d$, the application of several challenges to the same device can be expressed as a matrix-vector multiplication. For this, the challenges are collected as rows in a challenge matrix $\mathbf{C}$ and the delay differences are collected in a vector $\delta\mathbf{d}$. The multiplication of the challenge matrix and the delay vector $\mathbf{C} \cdot \delta\mathbf{d}$ results in a vector of analog secrets.

For the Arbiter PUF, the transformation from [12] can be used to obtain a similar linear model with $n+1$ variables. Here, the challenge vector C is replaced by the challenge parity vector P with $p_x = \prod_{i=x+1}^{n} c_i, p_n = 1$.

4 Model-Based Distinguishing Attack

While modeling attacks are already studied in various ways, we focus on mixed modeling and side-channel attacks, for scenarios where the output cannot be observed like for key-storage scenarios, the Interpose PUF [15], or the Multiplexer PUF [20]. For these scenarios, it is necessary to first train a model to match the observed leakage, which then can be used to predict the secret output.

4.1 Underlying Statistical Functions

The series of differential delay values on each device can be seen as a set of samples from a random distribution. The distribution is usually assumed to be a normal distribution $\mathcal{N}(\mu_0, {\sigma_d}^2)$. By the rules for distributions, the secret analog values would then also form a normal distribution $\mathcal{N}(n \cdot \mu_0, n \cdot {\sigma_d}^2)$.

Due to the measurement noise, it is impossible to accurately measure s in Eq. (1) even with unrestricted access to the device. Instead, only the noisy analog response $\tilde{s}$ is measured. In the general case, this noise is difficult to describe since it is a result of environmental conditions, phase jitter and possibly other semiconductor-specific effects. However, in this attack scenario hardware access is required for a side-channel analysis, and as such the attacker can usually control the environment and average over several traces. Without other influence, the noise from the frequency-based measurement tools is well approximated as Additive White Gaussian Noise (AWGN). Under this noise model, $\tilde{s} = s + e$, where the error value e is a sample from a normal distribution $\mathcal{N}(0, {\sigma_e}^2)$.

In typical SCA attacks on digital circuits, there would be a discrete number of hypotheses, thus, each can be assigned a probability of matching the measurement. However, this is not possible for continuous values since with infinitely many hypotheses the absolute probability of each converges to zero. Instead, probability density or likelihood can be used to obtain a relative comparison: Results "close" to the expectation μ are much more likely than results several standard deviations σ away. The probability density function (PDF) of a random distribution assigns each sample drawn from it to a likelihood. For a normal

distribution, the PDF is given by

$$\mathcal{P}(X = x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

The joint probability of several uncorrelated variables is given by the product of the marginal probabilities. Since this has unfavorable numerical properties, it is often replaced by the log-probability. For this, the natural logarithm is applied to each PDF, and the joint log-probability becomes the sum of the log-probabilities. For a normal distribution, this logarithm is

$$log\mathcal{P} = -\frac{1}{2\sigma^2}(x-\mu)^2 + \text{const}$$

This property is the underlying function of the least squares method used below: For a vector of normally distributed values, it can be proven that the one with the smallest sum over the squared distance to the mean constitutes the maximum likelihood. For other normal-like distributions, this method still poses a reasonable approximation.

4.2 Learning Strategy

To learn the secret delay differences δd of the linear model, the following steps are applied: We define hypotheses vectors h with elements $h_x \in \{-1, 1\}$ as a hypothesis for the signs that the differential analog responses s exhibit under the applied challenges in $\mathbf{C}$. Given the publicly known challenges $\mathbf{C}$ and the absolute magnitudes of the analog responses $|\tilde{s}|$ obtained from side channel measurements, the linear model leads for one specific h to

$$\left[\mathbf{C}\right] \cdot \delta d = \left[h \circ |\tilde{s}|\right] = \left[h \circ |s + e|\right],$$

with $\circ$ the element-wise product of two vectors and $|\bullet|$ the element-wise absolute value. With h, $|\tilde{s}|$, and $\mathbf{C}$ fixed and δd the variables, the following least squares problem is formulated:

$$\|e\|^2 = \min_{\delta d \in \mathbb{R}^n} \left\| \left[h \circ |\tilde{s}|\right] - \left[\mathbf{C}\right] \cdot \delta d \right\|^2. \tag{2}$$

The results of this minimization are (i) the most likely delay values $\delta\hat{d}$ (as the argument of the minimization) under the assumption that h is correct and (ii) an estimated least squares error, which is a form of likelihood that $\delta\hat{d}$ occurs under the assumption that h is correct.[1]

Since the correct hypothesis h is not known from the beginning, we use in Section 5 a machine-learning approach to iteratively approach likely values for h and the least squares optimization in Eq. (2) and the later Eq. (3) to compute the models $\delta\hat{d}$ under these hypotheses as well as the fitness $\|e\|^2$ of these hypotheses.

[1] Note that the exponential squared error of the solution formally does not represent an absolute probability of the hypothesis without adequate normalization. The correct mathematical term might be quasi-likelihood with an interpretation similar to a correlation coefficient.

4.3 Improved Likelihood Distinguishers

The fitness function in Eq. (2) poses a sufficient method when the number of measured challenges is large compared to the number of variables in the model. However, this function does not behave well when the number of measurements is low and the $\mathbf{C}$ matrix becomes almost square. In this situation, incorrect solutions of the type $\delta d = \mathbf{C}^{-1} \cdot \tilde{s}$ with no or almost no error exist. This underestimation of errors leads to a significant overestimation of some delay elements, breaking the likelihood estimation. Thus, while the leakage might still be large enough, the model will become incorrect. For our attack, we want to find a model that not only reproduces the observed measurements with a reasonable noise, but which is also likely to exist as a physical device in the first place.

This problem can be solved by also taking the expected distribution of δd into account. Unlike the (Deep) Neural Network models often used in attacks on PUFs like [21], the linear model directly represents physical properties. As such, it is possible to judge based on the delay weights whether or not the model can realistically correspond to a physical device. Assuming normal distribution of δd over all physical devices, the likelihood of observing a given vector of delays is again proportional to $\|\mu_0 - \delta d\|^2$. The individual delay elements can be represented by the identity matrix $\mathbf{I}$, so the previous equation can be extended to

$$\begin{bmatrix} \mathbf{C} \\ \mathbf{I} \end{bmatrix} \cdot \delta d = \begin{bmatrix} h \circ |\tilde{s}| \\ \mu_0 \end{bmatrix}$$

When optimizing for a low distance between model and side-channel measurement (first row) and for most likely delays (second row) at the same time, the two targets contradict each other As such, this system can only have solutions with a significant residual error. However, the error of the rows is not in the same domain, yet: The distribution of the noise on the measurements has a different variance than the distribution of the delay elements. In order for the sum over all squared errors to still represent a joint likelihood value for the model, the rows need to be weighted depending on the expected variance. The optimal weights can be derived from the coefficient $1/(2\sigma^2)$ in the log-PDF of the normal distribution: the squared error of each of the challenge equations is weighted by $1/(2\sigma_e^2)$, and the variance of each element is weighted by $1/(2\sigma_d^2)$, resulting in a weighted least squares equation:

$$\min_{\delta d \in \mathbb{R}^n} \left\| \begin{bmatrix} \frac{1}{2{\sigma_e}^2} \\ \frac{1}{2{\sigma_d}^2} \end{bmatrix} \circ \left(\begin{bmatrix} h \circ |\tilde{s}| \\ \mu_0 \end{bmatrix} - \begin{bmatrix} \mathbf{C} \\ \mathbf{I} \end{bmatrix} \cdot \delta d \right)^2 \right\| \tag{3}$$

It can be noted that for a large number of challenges or highly accurate measurements, the influence of all identity rows will converge to zero as their relative weights become small for all reasonable delay values. As such, the advantage of this improvement mostly comes into effect in unfavorable attack conditions.

Since for the attack a relative comparison of the likelihood values is sufficient, only the ratio of ${\sigma_d}^2$ to ${\sigma_e}^2$ has to be known or guessed. For profiled attacks, the

relation can be calculated directly. For non-profiled attacks, a rough estimation from the traces is possible: The variance of the noise can be estimated when m multiple measurements are taken for each value of s. The variance of a single measurement for a specific s is estimated as ${\sigma_m}^2 = 1/m \sum_{i=1}^{m} (s_{mi} - \mu)^2$, and after averaging over the measurements a noise of ${\sigma_e}^2 = {\sigma_m}^2/m$ remains. The variance of the delay elements ${\sigma_d}^2$ can also be estimated from the obtained measurements. For example, when $\mu_0 \approx 0$, it is approximated by averaging $(|\tilde{s}|^2/n)$ over all challenge measurements. For Arbiter PUF models, it should be noted that the element positions typically have different variances due to the start and arbitration gate, as well as signal edge degradation along the path.

4.4 Visualization of the Attack Based on Likelihood Distinguishers

The attack is visualized for a simulated ideal Loop PUF[2] in Fig. 5. For this, a ring with 8 stages is evaluated with a set of 16 pseudo-random challenges, which does not contain duplicated or inverse challenges. The violin plots show one sample; the exact shape can vary for each device. The secret analog value is overlaid with Gaussian noise of 2% and 50% of its variance, representing an SNR of 50 and 2. The quasi-likelihood as the exponential mean squared error is then calculated for all 2^{16} hypotheses. Since this is a linear model, each solution has an inverse model of $-\delta d$ corresponding to the complementary hypotheses $\neg h$. As such, there will always be one bit of entropy remaining to decide between the two models.[3]

For the low-noise scenario, it can be observed that there is a high correlation between the Hamming Distance to the correct or inverse hypothesis and the likelihood. The majority of the wrong hypotheses naturally have a Hamming Distance around 50% and a near-zero likelihood. The hypotheses with just a few bit errors on average reach a much higher likelihood, but there is a large spread in the individual values. This is due to the different impact of the bit-flip positions: A sign flip on an output with near-zero magnitude (corresponding to an unstable bit) results in almost the same model, but on a highly stable bit, the resulting model becomes implausible. When trying the hypothesis as the secret, starting from the one with the highest likelihood as a rank, we obtain the rank of the secret. On this device we can see a rank of 5.5, over many devices the median rank is 3 and 95% of devices are below 37. This is a drastic reduction of entropy since guessing the secret would take on average 2^{15} attempts.

With higher noise, the correlation with the Hamming Distance becomes much weaker. A larger portion of the hypotheses with several bit errors beats the correct hypothesis. Nevertheless, the rank with a median of 236 and 95% devices below 5960 still provides a significant reduction in guessing entropy.

The exhaustive method used in this visualization is only highly effective when the secret is provided by many small Loop PUF units. However, the number of

[2] We call here a Loop PUF with delay elements chosen from a zero-mean normal distribution, which suffers only from normally distributed noise, "ideal".

[3] In other words, one bit of h could be fixed reducing the hypothesis space, but always h and $\neg h$ have to be considered as possible solutions.

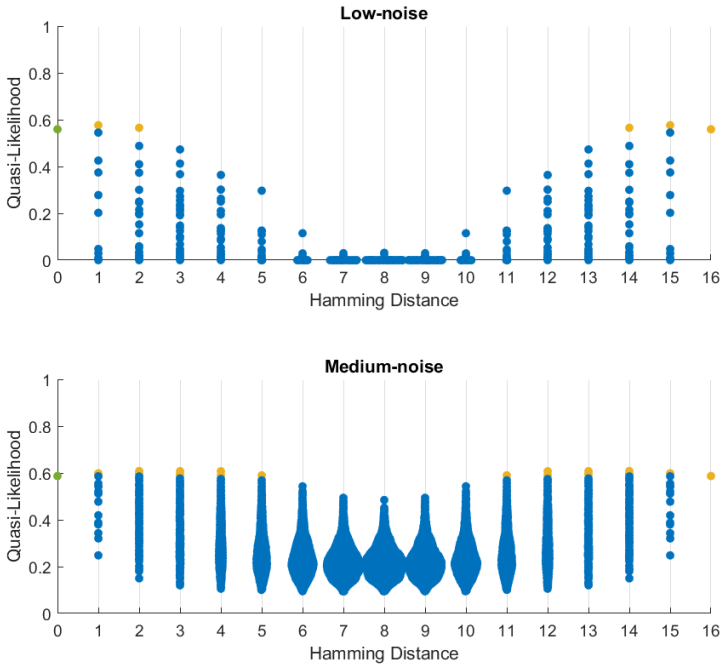

Fig. 5: Likelihood of all 2^{16} hypothesis on a 8-stage ring at 2% (low-noise) and 50% (medium-noise) noise variance. Yellow dots represent false guesses with a higher value than the correct green dot at the very left. The Hamming Distance is the distance to the correct guess.

hypotheses might be much larger in practice, rendering this exhaustive approach infeasible. The practical attack will be discussed in the next section.

4.5 Underdetermined Systems

Given that the model of an n-stage Loop PUF has n variables, it takes at least n linearly independent challenges and the corresponding signed secrets s to build a correct model of a device. In practical designs, it would be desired to obtain as many bits as possible from a small PUF. Under this condition, the system of equations is expected to be overdetermined.

However, an overdetermined equation system is not necessary to use the attack described in this work. When fewer challenges are used than there are stages or when not all challenges can be measured for some reason, this would lead to an underdetermined system with more unknown delay values to guess than available measurements. Such a system has infinitely many solutions for every hypothesis with zero error and thus would not yield any results. But when Eq. (3) is used, the solution with the lowest variance on the delay elements will be picked. While this could never result in an ideal model, it can already exclude

some of the hypotheses as unlikely because they would require unrealistically large elements even in the best case.

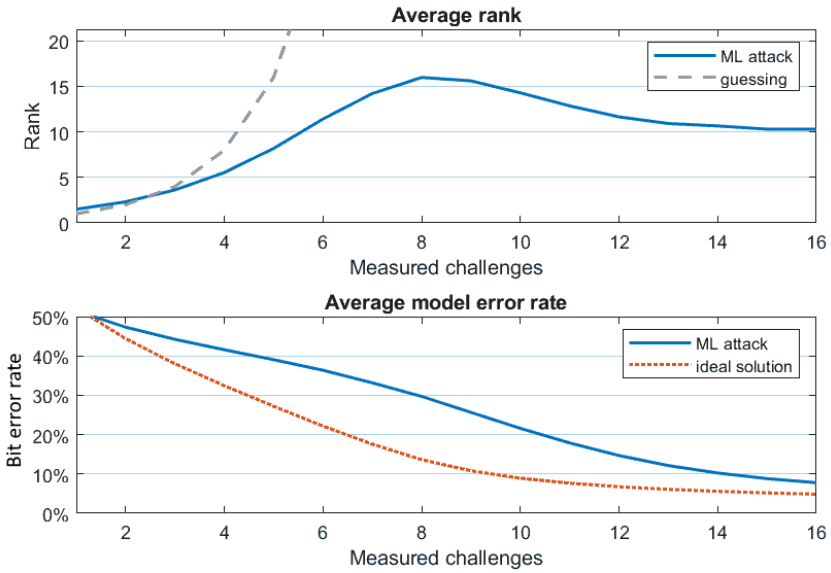

Fig. 6: Average rank and model accuracy under the low noise scenario. The average rank is already better than the guessing entropy when the system is still underdetermined for up to 7 challenges.

This is visualized in Fig. 6 with the same low noise setting as the previous figure. Here, the number of available measurements from pseudorandom challenges is varied, and the average over 100k virtual devices is obtained by a Monte-Carlo simulation. The average rank of this attack is already better than guessing, even for less than 8 measurements. However, the model accuracy remains low until a larger number of measurements is obtained. Even when the signed noisy analog values are used for the model ("ideal solution" in the figure), the error rate of the model will only slowly improve due to the noise and poor distribution of the challenges.

4.6 MAP Estimation

It should be noted that the attack strategy presented in this work can be easily expanded to include prior information to form a maximum a posteriori (MAP) estimator. Practical implementations of PUF designs will always result in some imperfections compared to this ideal model. Even if the design itself is perfectly symmetrical, there are various semiconductor-specific effects which are out of control of the designer. This is well known for the influence of parasitics and coupling capacities: The other components placed on the chip as well as the

power grid are not symmetric around the PUF. When this increases, e.g., the effective capacity of a wire in the PUF's delay line, the corresponding path will constantly have a higher delay on all devices.

When an attacker has already broken several identical devices, they might have a more accurate estimation of the distribution of each delay element. This would mostly lead to different values of μ for each element, which can then replace the idealized value of $\mu_0 = 0$ in the equation system. In more advanced analysis, there might also appear a correlation between delay elements. In this case, the variance vector could be replaced by a covariance matrix, forming a generalized least squares problem. The complexity of solving such systems and the rest of this attack remain similar.

Under this imperfection of real implementations, attacks on ideal PUFs present the most difficult case. As such, the results of ideal simulations should never be confused with a bound of attacker capabilities.

5 Practical Attacks

After the theoretical considerations, this section provides a strategy to apply the attack to larger hypothesis spaces. Afterward, it shows the result of a practical attack and finally introduces an improved attack using helper data leakage in case of an attack on a PUF-based key storage.

5.1 Attacks with Large Hypotheses Spaces

The basic attack introduced before needs to test all hypothesis by calculating each likelihood model. For the least squares operation itself, highly efficient numerical algorithms like LSQR [17] exist: the computational complexity only grows moderately with the number of PUF stages n or challenge-stability pairs m, typically $O(3m + 5n)$ for each of a small number of iterations. However, the number of required models grows exponentially with the number of used challenge-stability pairs, resulting in an effective complexity of $O(2^m)$.

In case of a long output sequences per PUF, it is not feasible to find the hypothesis with the highest likelihood by an exhaustive search. But as shown in Figure 5, there is a general correlation between the Hamming Distance and the likelihood. As such, when a tested hypothesis only yields a low value, it is highly unlikely that it is close to the desired solution. However, this does not pose a smooth gradient, even hypotheses with a few bit flips can sometimes score almost zero likelihood.

This structure of the binary search space narrows down the available search algorithms, which could still extract some high-likelihood solutions with non-exhaustive complexity. One method, which still performs well under such settings, is Genetic Algorithm (GA). For this, random starting points are selected in the vector space. From there on, pairs of the most likely hypotheses are mixed by keeping the equal bits and randomly selecting the unequal ones. Mutations of low Hamming Distance and new random vectors are added to the pool to prevent

the algorithm from converging too fast. One output bit should be permanently fixed because of the inverse model. This not only cuts the search space in half, it also prevents simultaneous convergence against both solutions (cf. Section 4.4).

The proposed genetic algorithm converges quickly to a highly likely hypothesis (typically in our experiments we observed convergence within 100 generations for hypotheses up to 256 bits and a population of 3k). This allows to cut down the computation time of this attack to a few minutes on desktop CPUs.

Larger systems become inherently more sensitive to noise. As such, even under the low noise scenario, the most likely hypothesis will usually be several bits off from the correct solution. As a consequence, it is preferred to find a list of likely candidates containing the correct solution rather than opting for finding only a single good solution. A good approach is to adapt the algorithm to obtain a large population of good solutions and to brute-force bit positions with low stability in the obtained model. But since PUFs are not error-free anyway, even approximate solutions might be acceptable: Current fuzzy extractors, e.g., allow for error correction to derive the actual key. Depending on the implementation, it can correct about 5% to 15% or even 25% of errors directly, allowing for the same error in the prediction from the PUF model.

5.2 Attack Results for a Practical PUF Implementation

The device of Figure 2 consists of a 32-stage Loop PUF and uses the temporal masking scheme with randomness obtained from the PUF oscillator. The challenges are generated by a 32-bit Xorshift pseudorandom number generator [13] with a fixed seed. The cycle length already prevents duplicates, but to prevent inverse or low-distance pairs of challenges, it is necessary to pick an adequate seed.

The unsigned stability is extracted from 200 power side channel traces at 1.25 GSa/s, covering the entire runtime of the PUF. For this, the power spectrum curves of the time slots corresponding to a pair of challenge and challenge complement are obtained by averaging over the STFT time segments. Then, a peak detection is used to find the center of the frequency. This method is highly accurate and results in only 0.3% noise variance. Since the first dummy round is unmasked, one bit of the system is already known. The ES algorithm returns the optimal solution, which only has a Hamming Distance of 2 to the correct PUF output. This amount of bit errors also has to be expected from the physical device under environmental effects, as such the accuracy of the obtained model lies well in the reliability of Loop PUF devices.

The resulting model can be seen in Fig. 7, compared to the correct model derived from the counter values. After normalization for comparison, the guessed delay values δd are very close to the ideal model. It can be noticed that the differential delay elements are heavily biased (nearly all differences are positive), but this did not reduce the effectiveness of this attack.[4]

[4] Please note that a bias in the differential delay is expected for the Loop PUF, since all delay elements are implemented the same way without compensation for bias and bias in the overall PUF response is only compensated by using certain challenges.

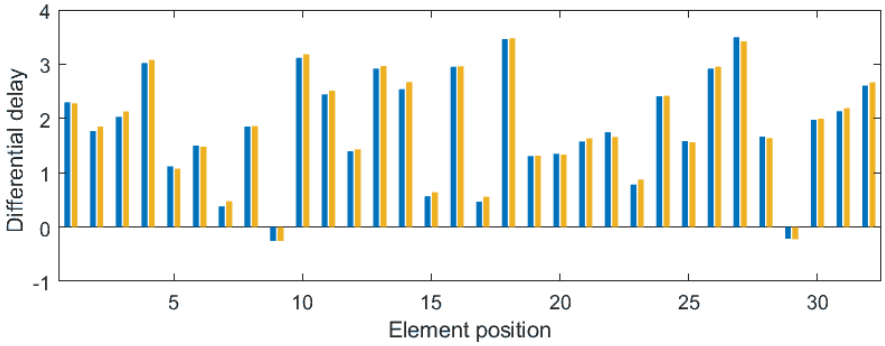

Fig. 7: The differential delay values of the test device reconstructed from the attack (blue) vs. the correct model from counter values (yellow) after normalization to standard deviation 1.

Similar results arise on the simulated 64-stage Arbiter PUFs: the relation between the stability and settling time was fitted with a two-term exponential function and 5 devices were simulated without noise. All could be successfully attacked after 128 challenges, resulting in 0-2 bit errors and numerically close models.

5.3 Amplifying Helper Data Leakage

When the PUF is used in a key storage scenario, the secret output will be processed further to remove the bit errors. For this, an error-correcting code (ECC) is necessary, which will encode the secret key. A helper data algorithms like the Fuzzy Commitment scheme [8] is used to encapsulate the key, resulting in only non-secret data needing to be stored. However, such schemes are not cryptographically strong, so the helper data might still leak a small amount of entropy. It was already discussed in [22] that this leakage might be sufficient to break a PUF with challenge-response behavior if a high number of challenges is used. The leakage can also be considered in the attack in this work to obtain a solution more efficiently.

Only codewords of the ECC (or more precisely, frequently a mapping of codewords via helper data) would pose valid solutions of the unknown sign vector. Because of this redundant information, the search space for the hypothesis can be reduced significantly to the message length of the code, which is typically the key length or a fraction of it. For each message hypothesis h' used in the search algorithm, the corresponding sign vector $h = \mathcal{F}(h', \text{helperdata})$ is used in the likelihood estimator. For the Fuzzy Commitment, this function is $h = \text{enc}(h') \oplus \text{helperdata}$.

The performance of the GA algorithm on this problem highly depends on the used error correction code. For repetition codes or other codes with a sparse generator matrix, a small change in the message h' only results in a small change in the codeword and h. As such, the crossover and mutation of h' vectors still

produces a h vector with the expected fitness. But for large block codes like the commonly used Bose-Chaudhuri-Hocquenghem (BCH) code, this is no longer the case. A single bit flip in the message h' will affect half of all the parity bits and thus a significant part of h. This can prevent convergence of the algorithm since it is not possible to obtain low-distance h vectors by evolution of the message.

Often, fuzzy extractors are constructed with concatenated codes (such as a repetition code) as inner code and a BCH code as outer code. This offers the elegant solution of attacking the intermediate values by the GA algorithm, and then to use the decoder of the outer code to resolve the few remaining bit errors.

So far, only attacks on individual units have been considered. But when Loop PUFs are used with an inner code with small blocks, a reasonable design choice is to operate several PUFs in parallel to match the word size. A realistic implementation, as demonstrated, for example, in [10], would be 5 Loop PUFs serially feeding 5x 255 bits each to a 5-Repetition decoder, and the remaining 255 bits of intermediate values are decoded by a [255,128,19] BCH decoder to obtain the 128 bit key. This results in a significant speed-up of the reconstruction while reducing the amount of data that has to be stored in the decoder. In this scenario, it would be sufficient to only attack one of the Loop PUFs, but this does not make use of all available information. Instead, all devices can be attacked in parallel by solving a system of disconnected devices. For this, the vectors of delay elements are concatenated and the challenge and identity matrix are expanded by blocks of zeros.

$$\begin{bmatrix} \mathbf{C} & \dots & \mathbf{0} \\ \vdots & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C} \\ \mathbf{I} & \dots & \mathbf{0} \\ \vdots & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \delta d_1 \\ \vdots \\ \delta d_n \end{bmatrix} = \begin{bmatrix} h \circ |\tilde{s}| \\ \mu_0 \end{bmatrix}$$

This combined attack was tested on 100 simulated devices with 5 Loop PUFs of 128 stages and 255 challenges. The attack could directly break 100% of devices under the low 2% noise variance with an average of 3.6 bit errors before the outer decoder. Under the medium 50% noise variance, it was still possible to break 98% of devices with an average 8.7 bit errors before the decoder. Only for two devices, the number of errors exceeded the correction capability of 19 bits for the BCH code.

6 Countermeasures and Discussion

Various ways to increase the complexity of the attack in this work exist. However, the desired countermeasure should solve the root cause of the leakage that makes this attack possible: the poor structure of pseudorandom challenge sets which induces some correlation in the analog secrets. Only for a set of orthogonal challenges as given by a Hadamard matrix, all secret bits have full entropy. Under

this condition, all hypotheses are equally probable without further knowledge of any bias. As shown in Figure 8, this attack becomes impossible regardless of the noise level. This is already implemented by some designs in respect to other attacks, but the reduced number of challenges also reduces the bits obtained per area.

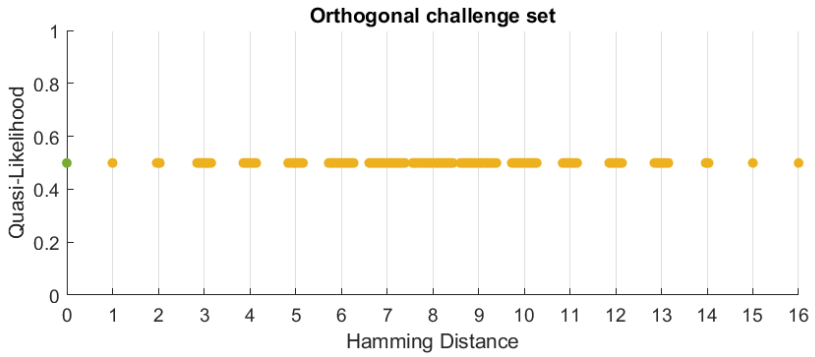

Fig. 8: The likelihood for all hypotheses is equal when an orthogonal challenge set is used on an ideal PUF.

It is, however, an important finding that the proposed expansion of this challenge set in [18] does not behave as expected and does not provide the expected amount of entropy under our attack scenario. As shown in Figure 9, the attack instantly becomes better than guessing again as soon as one non-orthogonal challenge is added to the orthogonal challenge set. The model accuracy actually becomes slightly better under such a set since the challenges are more evenly distributed. This does not only put a very low limit of n on the number of secure challenges. It also shows the danger of fault injection in such a Hadamard challenge generator. With just a few targeted injections repeated for a few measurements, every secure but not otherwise protected design could be compromised.

Other straightforward countermeasures against the presented modeling attack would still suffer from vulnerability to either the attack in this work or other existing attacks: A true shuffling would prevent the leakage of unsigned stability, but still becomes vulnerable to entropy-based attacks in [22] when significantly too many bits are extracted. An attack based on Helper Data leakage can be hindered or prevented by the choice of the Fuzzy Extractor and error correcting code. However, the inclusion of Helper Data leakage is in general not necessary for the attack in this work.

Similarly, stronger nonlinear PUF constructions like Interpose PUFs or XOR constructions could be used. While these are significantly harder to model based on the output or Helper Data leakage, the complexity of side-channel attacks can stay low: As long as the unsigned stability (e.g. frequency) of the individual elements can be separated by localized analysis, the remaining entropy will be

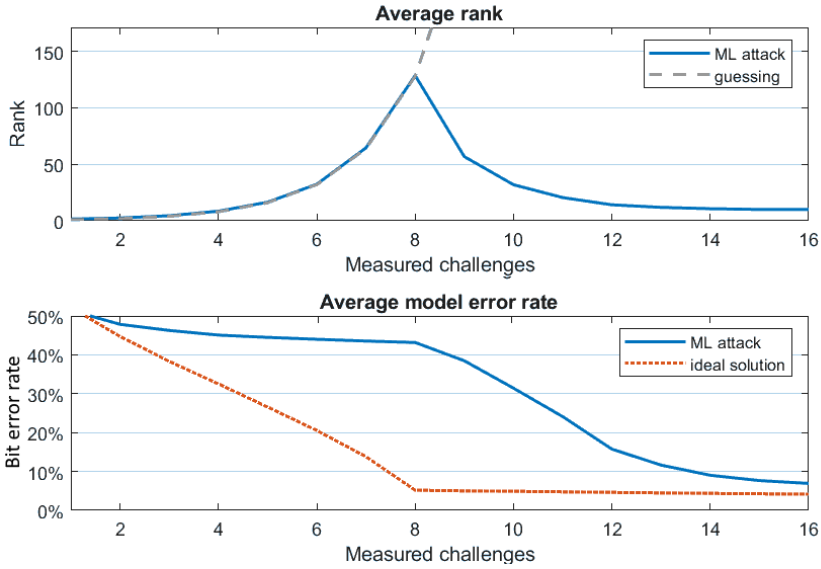

Fig. 9: An ideal challenge set on an 8-stage ring. The first 8 orthogonal challenges are not vulnerable, but the addition of even one more challenge breaks the system.

small for reasonable-sized designs. Such localized measurements, however, have been proven to be possible at least on FPGAs [14].

7 Conclusion

This work brought together side channel analysis attacks and machine learning of PUFs and has demonstrated the necessity to limit the number of challenges in PUF scenarios where reliability information can be obtained, e.g., through a side-channel analysis. Even if no challenge-response pairs can be collected and the actual secret remains hidden from an attacker, a model can be trained to significantly reduce the entropy and ultimately break the construction: The leakage of unsigned stability allows for the prediction of PUF structures that have a linear model, and model-based distinguishing is possible even under noisy measurements. The work has confirmed the applicability of such an attack under numerical PUF models as well as using an actual side-channel attack on a Loop PUF.

To counter the attack, either the use of an orthogonal set of challenge vectors or a true shuffling method can be used to protect the secret of a PUF. Both methods, however, result in a significantly higher area per bit and, thus, reduce the cost advantage of the PUF design. Nevertheless, they still allow for a secure implementation.

Acknowledgments. This work was partly funded by the Deutsche Forschungsgemeinschaft (DFG) in the project STAMPS-PLUS (grant no. PE 3242/2-2).

References

1. Aghaie, A., Moradi, A.: TI-PUF: Toward Side-Channel Resistant Physical Unclonable Functions. IEEE Transactions on Information Forensics and Security **15**, 3470–3481 (2020). https://doi.org/10.1109/TIFS.2020.2986887
2. Becker, G.T.: The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs. In: Güneysu, T., Handschuh, H. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2015. pp. 535–555. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
3. Chen, C., Veldhuis, R.N.J., Kevenaar, T.A.M., Akkermans, A.H.M.: Multi-Bits Biometric String Generation Based on the Likelihood Ratio. In: 2007 First IEEE International Conference on Biometrics: Theory, Applications, and Systems. pp. 1–6 (2007). https://doi.org/10.1109/BTAS.2007.4401912
4. Cherif, Z., Danger, J.L., Guilley, S., Bossuet, L.: An Easy-to-Design PUF Based on a Single Oscillator: The Loop PUF. In: 2012 15th Euromicro Conference on Digital System Design. pp. 156–162 (Sep 2012). https://doi.org/10.1109/DSD.2012.22
5. Danger, J.L., Guilley, S., Schaub, A.: Two-Metric Helper Data for Highly Robust and Secure Delay PUFs. In: 2019 IEEE 8th International Workshop on Advances in Sensors and Interfaces (IWASI). pp. 184–188 (2019). https://doi.org/10.1109/IWASI.2019.8791249
6. 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. p. 148–160. CCS '02, Association for Computing Machinery, New York, NY, USA (2002). https://doi.org/10.1145/586110.586132
7. de Groot, J.A., Skoric, B., de Vreede, N., Linnartz, J.M.G.: Quantization in Zero Leakage Helper Data Schemes. EURASIP Journal on Advances in Signal Processing **2016**, 54 (2016). https://doi.org/10.1186/s13634-016-0353-z
8. Juels, A., Wattenberg, M.: A Fuzzy Commitment Scheme. In: ACM Conference on Computer and Communications Security (CCS). pp. 28–36. ACM (1999)
9. Kestel, C., Frisch, C., Pehl, M., Wehn, N.: Towards more secure puf applications: A low-area polar decoder implementation. In: 2022 IEEE 35th International System-on-Chip Conference (SOCC). pp. 1–6. IEEE (Sep 2022). https://doi.org/10.1109/socc56010.2022.9908130
10. Kroeger, T., Cheng, W., Guilley, S., Danger, J.L., Karimi, N.: Cross-PUF Attacks on Arbiter-PUFs through their Power Side-Channel. In: 2020 IEEE International Test Conference (ITC). pp. 1–5 (2020). https://doi.org/10.1109/ITC44778.2020.9325241
11. Kroeger, T., Cheng, W., Guilley, S., Danger, J.L., Karimi, N.: Assessment and mitigation of power side-channel-based cross-puf attacks on arbiter-pufs and their derivatives. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **30**(2), 187–200 (2022). https://doi.org/10.1109/TVLSI.2021.3129141
12. Lim, D.: Extracting Secret Keys from Integrated Circuits. Massachusetts Institute of Technology (May 2004), master Thesis
13. Marsaglia, G.: Xorshift RNGs. Journal of Statistical Software **8**(14), 1–6 (2003). https://doi.org/10.18637/jss.v008.i14, `https://www.jstatsoft.org/index.php/jss/article/view/v008i14`
14. Merli, D., Heyszl, J., Heinz, B., Schuster, D., Stumpf, F., Sigl, G.: Localized Electromagnetic Analysis of RO PUFs. In: Proceedings of the IEEE Int. Symposium of Hardware-Oriented Security and Trust. IEEE (Jun 2013)

15. Nguyen, P.H., Sahoo, D.P., Jin, C., Mahmood, K., Rührmair, U., van Dijk, M.: The Interpose PUF: Secure PUF Design against State-of-the-art Machine Learning Attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems **2019**(4), 243–290 (Aug 2019). https://doi.org/10.13154/tches.v2019.i4.243-290, https://tches.iacr.org/index.php/TCHES/article/view/8351
16. Nöpel, J., Music, T., Stein, N., Frisch, C., Pehl, M.: Quantization schemes for pufs: The entropy-area trade-off. In: 2025 IEEE International Symposium on Hardware Oriented Security and Trust (HOST) (2025)
17. Paige, C.C., Saunders, M.A.: Lsqr: An algorithm for sparse linear equations and sparse least squares. ACM Trans. Math. Soft. **8**, 43–71 (1982)
18. Rioul, O., Solé, P., Guilley, S., Danger, J.L.: On the entropy of physically unclonable functions. In: 2016 IEEE International Symposium on Information Theory (ISIT). pp. 2928–2932 (2016). https://doi.org/10.1109/ISIT.2016.7541835
19. 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 '10, ACM, New York, NY, USA (2010). https://doi.org/10.1145/1866307.1866335
20. Sahoo, D.P., Mukhopadhyay, D., Chakraborty, R.S., Nguyen, P.H.: A Multiplexer-Based Arbiter PUF Composition with Enhanced Reliability and Security. IEEE Transactions on Computers **67**(3), 403–417 (2018). https://doi.org/10.1109/TC.2017.2749226
21. Santikellur, P., Bhattacharyay, A., Chakraborty, R.S.: Deep Learning Based Model Building Attacks on Arbiter PUF Compositions. Cryptology ePrint Archive, Paper p. 566 (2019), https://eprint.iacr.org/2019/566
22. Strieder, E., Frisch, C., Pehl, M.: Machine learning of physical unclonable functions using helper data: Revealing a pitfall in the fuzzy commitment scheme. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(2), 1–36 (Feb 2021). https://doi.org/10.46586/tches.v2021.i2.1-36, https://tches.iacr.org/index.php/TCHES/article/view/8786
23. Suh, G.E., Devadas, S.: Physical Unclonable Functions for Device Authentication and Secret Key Generation. In: ACM/IEEE Design Automation Conference (DAC). pp. 9–14 (2007)
24. Tebelmann, L., Danger, J.L., Pehl, M.: Self-Secured PUF: Protecting the Loop PUF by Masking. In: International Workshop on Constructive Side-Channel Analysis and Secure Design. pp. 293–314. Springer (2020)
25. Tebelmann, L., Kühne, U., Danger, J.L., Pehl, M.: Analysis and Protection of the Two-Metric Helper Data Scheme. In: International Workshop on Constructive Side-Channel Analysis and Secure Design. pp. 279–302. Springer (2021)
26. Tebelmann, L., Pehl, M., Immler, V.: Side-Channel Analysis of the TERO PUF. In: Polian, I., Stöttinger, M. (eds.) Constructive Side-Channel Analysis and Secure Design. pp. 43–60. Springer International Publishing, Cham (2019)
27. Tebelmann, L., Wettermann, M., Pehl, M.: On-Chip Side-Channel Analysis of the Loop PUF. In: Proceedings of the 2022 Workshop on Attacks and Solutions in Hardware Security. pp. 55 – 63. ASHES'22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3560834.3563827
28. Xu, X., Rührmair, U., Holcomb, D.E., Burleson, W.: Security evaluation and enhancement of bistable ring pufs. In: Mangard, S., Schaumont, P. (eds.) Radio Frequency Identification. pp. 3–16. Springer International Publishing, Cham (2015)