Analysis and Protection of the Two-metric Helper Data Scheme*

Lars Tebelmann[1][0000−0003−2014−7184], Ulrich Kühne[2][0000−0002−0855−8223], Jean-Luc Danger[2][0000−0001−5063−7964], and Michael Pehl[1][0000−0001−6100−7714]

[1] Technical University Munich, Munich, Germany
TUM Department of Electrical and Computer Engineering
Chair of Security in Information Technology
{lars.tebelmann,m.pehl}@tum.de
[2] Télécom Paris, Paris, France
{jean-luc.danger,ulrich.kuhne}@telecom-paris.fr

Abstract. To compensate for the poor reliability of Physical Unclonable Function (PUF) primitives, some low complexity solutions not requiring error-correcting codes (ECC) have been proposed. One simple method is to discard less reliable bits, which are indicated in the helper data stored inside the PUF. To avoid discarding bits, the Two-metric Helper Data (TMH) method, which particularly applies to oscillation-based PUFs, allows to keep all bits by using different metrics when deriving the PUF response. However, oscillation-based PUFs are sensitive to side-channel analysis (SCA) since the frequencies of the oscillations can be observed by current or electromagnetic measurements. This paper studies the security of PUFs using TMH in order to obtain both reliable and robust PUF responses. We show that PUFs using TMH are sensitive to SCA, but can be greatly improved by using temporal masking and adapted extraction metrics. In case of public helper data, an efficient protection requires the randomization of the measurement order. We study two different solutions, providing interesting insights into trade-offs between security and complexity.

Keywords: PUF, Side-Channel Analysis, Two-metric Helper Data, LFSR-based Protection, Permutation, Countermeasures

1 Introduction

Physical Unclonable Functions (PUFs) have become an important security primitive, which can greatly enhance authentication mechanisms in digital devices. A good PUF provides a unique identifier, which does not have to be programmed in non-volatile memory (NVM) and hence is not sensitive to memory hacking.

* This work was partly funded by the German Ministry of Education and Research in the project SecForCARs under grant number 01KIS0795 and under the SPARTA project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 830892.

This identifier corresponds to the value returned by a function that exploits slight technological differences from the fabrication process, and which stay constant during life time of a component. The extraction can be done in many different ways, such as measuring the delay – giving rise to the delay PUF [17] – and the initial state of SRAM cells – the SRAM PUF [7]. In particular, a PUF has the property of *unclonability*: the outputs depend on process variations only, i.e., copying a device from the same blueprint does not yield the same PUF response.

One of the main drawbacks of PUFs is the reliability, as the bit error rate (BER) of their output can be in the range of 3 to 15% [9]. The most common way to improve the reliability is the use of error-correcting codes (ECC) [6, 10]. It requires to generate a public word during the enrollment phase, termed helper data, which enables correction of faulty PUF bits during the reconstruction phase. Alternatively, the reliability can be improved if the PUF output contains an indication of the bit reliability. For example, oscillation-based PUFs such as the Ring Oscillator (RO) PUF [17] or Loop PUF [2] use the sign of a frequency difference as secret, while the difference's magnitude provides reliability information regarding the probability of bit flips. A first approach is to discard unreliable bits [15]. In this work, we examine an improved method, called the Two-metric Helper Data (TMH) scheme [3], that does not loose unreliable bits. By reconstructing under two different metrics, it can make the use of ECC unnecessary.

PUFs have been attacked by side-channel analysis (SCA) in different ways, e.g., targeting the post-processing stage [13, 19]. Oscillation-based PUFs have been attacked by observing the electromagnetic (EM) emanations from the RO PUF [11, 12, 16] and the Loop PUF [18]. To face this threat, protections based on randomization have been proposed, such as using a random path for the RO PUF [11] and temporal masking for the Loop PUF [18]. While helper data manipulation has been used as an attack vector [4, 5, 1], the impact of the helper data algorithm for SCA has been neglected so far.

Contribution: We show that the TMH scheme [3] is prone to yet unexplored SCA attacks. As a consequence, new protection mechanisms are needed. In particular:

1. We present a novel attack against the TMH scheme, which makes use of the *magnitude* – in contrast to the *sign* – of the frequency differences in oscillator-based PUFs. The feasibility of the attack is shown for a 63-bit Loop PUF.
2. We provide two different approaches for countermeasures that combine modification of the TMH metrics with randomization of the Loop PUF challenge order. We show trade-offs between security and cost.
3. We deeply analyze the security of the proposed solutions, showing that the low-complexity solution – while requiring less randomness – may be attacked, although with higher effort than without the protection.

Structure: The remainder of this work is organized as follows: Section 2 explains the TMH method. Sections 3 and 4 provide the threat model and the security

analysis of the TMH respectively. Protections are presented in Section 5, their security analysis in Section 6 and a conclusion is drawn in Section 7.

2 Two-metric Helper Data Method

The TMH is a method that enhances the reliability level of PUFs to generate secret responses used as cryptographic key [3]. It applies specially to delay PUFs, whose responses contain reliability information. Oscillator-based PUFs, such as the RO PUF [17] and the Loop PUF [2] use the sign of a frequency difference *df* as response bit. The magnitude of the difference provides reliability information. Using this information, the TMH method can allow for highly reliable responses without the use of ECC decoders in the post-processing stage.

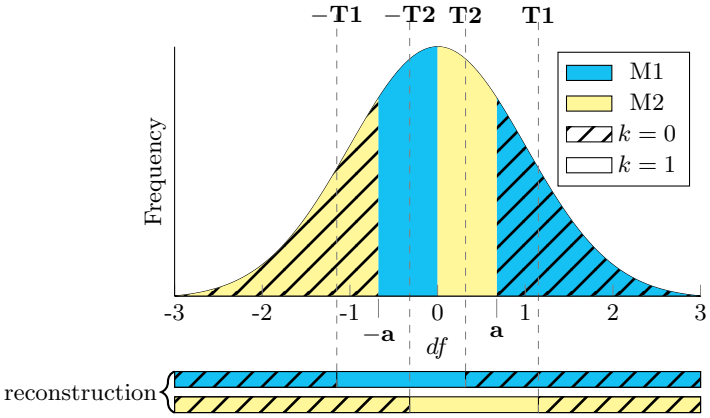


Fig. 1: Choice of metric and extraction of PUF bit value from the the frequency difference *df* according to [3].

For this purpose, in the *enrollment phase*, the distribution of all *df* on a device is estimated under the assumption that it follows a Gaussian normal distribution with mean $\mu = 0$ and variance σ^2, i.e., $df \sim \mathcal{N}(0, \sigma^2)$. The distribution is divided into octiles defined by the points $-T1$, $-a$, $-T2$, $T2$, a, $T1$ (and $\pm\infty$) as depicted in Fig. 1. For each octile the upper and lower bounds x and y are adapted such that, the cumulative distribution function (CDF) $\Phi(\cdot)$ defined by the integral over the probability density function (PDF) $\phi(\cdot)$ complies to

$$\Phi(x) - \Phi(y) = \int_y^x \phi(x)dx = \frac{1}{8}. \tag{1}$$

The Two-metric Helper Data (TMH) method derives its name from the fact that based on the octiles two metrics $M1$ and $M2$ define the mapping of the

frequency difference df to the PUF bit k as:

$$M1 : k = \begin{cases} 0, T2 \leq df \vee df < -T1 \\ 1, -T1 \leq df < T2 \end{cases} \quad M2 : k = \begin{cases} 0, T1 \leq df \vee df < -T2 \\ 1, -T2 \leq df < T1 \end{cases} . \quad (2)$$

Note that from the definition of the octiles $\int_{-T1}^{T2} \phi(x)dx = \int_{-T2}^{T1} \phi(x)dx = 1/2$, i.e., the values for k are equiprobable and no bias is induced by the TMH. For the frequency difference df_C of a particular challenge C the metric is chosen and stored as helper data w_C, for which the reconstruction from a perturbed $df'_C = df_C + \delta$ is more reliable. In other words, the metric is stored as helper data, for which a deviation δ from the enrollment value is less likely to cause a change of the PUF bit during reconstruction. From Fig. 1 metrics $M1$ and $M2$ are least stable around $-T1/T2$ and $-T2/T1$ respectively, thus the selection of the appropriate helper data is done according to the following intervals:

$$w_C = \begin{cases} M2, & -a < df_C \vee 0 < df_C \leq a \\ M1, & -a \leq df_C \leq 0 \vee a < df_C \end{cases} . \quad (3)$$

During the *reconstruction phase* the frequency difference df'_C is mapped with the metric stored in the helper data w_C to k'_C. As the bounds from the enrollment $\pm T1$, $\pm T2$ and $\pm a$ may change due to environmental conditions, the device estimates a new set $\pm T1'$, $\pm T2'$ and $\pm a'$ for reconstruction i.e.,

$$k'_C = \begin{cases} 1, & -T1' \leq df'_C < T2' \,\&\, M1 \vee -T2' \leq df'_C < T1' \,\&\, M2 \\ 0, & -T2' \geq df'_C \geq T1' \,\&\, M2 \vee -T1' \geq df'_C \geq T2' \,\&\, M1 \end{cases} . \quad (4)$$

For typical noise measurements a BER of $< 10^{-6}$ is achieved [3], i.e., $k'_C = k_C$ with high probability.

3 Attacker Model

Throughout the paper we adopt an attacker model where the attacker has physical access to the targeted device. In particular, the attacker has read access to the helper data of the PUF, i.e., it is known which metric $M1$ and $M2$ of the TMH scheme is used to derive a particular PUF bit. The assumption of public helper data is generally accepted when designing PUF-based key storage as (i) the helper data should not leak about the secret [14] and (ii) storing the helper data in a protected NVM would render the PUF approach useless as the secret could directly be stored. In theory, access to the helper data could be hampered, e.g. if it is stored by fuses, that are difficult to read out. Limited helper data represents the worst case scenario for an attacker, which is why we also investigate the possibility of an attack without helper data knowledge. However, the security of the system must not depend on the concealment of the helper data.

Furthermore, the attacker is able to perform passive physical attacks such as side-channel measurements during the reconstruction phase of the PUF. This includes power measurements as well as EM measurements above the package. We neglect semi-invasive attacks such as localized EM SCA, which have been

applied to RO PUFs [12, 11, 16], but require a high level of sophistication due to decapsulation of chips, and expensive measurement equipment in terms of micro near-field probes. The bar for mounting these kinds of attacks can be raised by adding sensors that detect opening of the package and thus impede direct access to the die. As we focus on sequentially evaluated PUFs like the Loop PUF, the application of semi-invasive means does not promise any advantage compared to non-invasive attacks that would justify the additional effort.

For all considerations, the attacker is able to acquire multiple measurements of the reconstruction phase for the same PUF device. That means for the countermeasures proposed in Section 5 that the attacker can compare measurements with different randomization to establish a relationship among measurements. The number of measurements can practically be limited by the time and storage capacities of the adversary, but we assume no countermeasures to limit the number of reconstructions.

4 Analysis of the Two-metric Helper Data Method

In this section, we describe a side-channel vulnerability of the original TMH method given an attacker without helper data access. We propose a modification of the TMH from Section 2 that improves the scheme regarding mitigation of the vulnerability. Subsequently, we show that even with the improved method an attack with helper data knowledge still succeeds in recovering the secret. The findings emphasize the need for additional countermeasures, which are proposed in Section 5.

4.1 SCA Attack Vector of the Two-metric Helper Data Method

For oscillation-based PUFs, the use of the sign of frequency difference df can be targeted by side-channel analysis. In case of the RO PUF, attacks target the comparison of frequencies [12] and the resolution of single oscillators or counters using localized EM measurements [11]. Even if components are placed close to each other, EM attacks succeed due to *geometric leaks* [16], making an effective protection against SCA difficult.

A possible improvement regarding side-channel resistance is the so-called Loop PUF [2] that uses a single RO configured by challenges. In order to generate a PUF bit k, a challenge C and its complement $\neg C$, where each bit is flipped, generate the frequencies f_C and $f_{\neg C}$. The sign of the difference $df = f_C - f_{\neg C}$ is then taken as the PUF bit. While the original Loop PUF design is vulnerable to SCA, because the frequencies can be measured sequentially, a simple and effective countermeasure called *temporal masking* can be applied [18]. It randomizes the order of the two challenges to derive df, therefore an attacker is not able to deduce the sign of df anymore. The required randomness is derived from the counter's least significant bit (LSB), in other words from the oscillators phase jitter, enabling a self-protected PUF design.

In the following and without loss of generality, we assume that the frequency differences df processed by the TMH stem from a Loop PUF that is protected by

the *temporal masking* scheme. With this approach the underlying PUF primitive is protected against SCA and we can focus on the properties of the TMH. First, note that for the TMH method the bit value is not given by the sign of *df* but by its magnitude. Revisiting Fig. 1 and Eqs. (2) and (3), which establish the bit value according to the metrics and *df*, if the absolute value of *df* is greater than the threshold a, the bit value is always $k = 0$. Furthermore for absolute values of *df* around 0, the bit value is $k = 1$. Thus an attacker observing the frequencies and their difference in the side-channel can easily derive the bit value even in presence of the temporal masking countermeasure. This is only possible because the TMH uses the magnitude instead of the sign to derive the PUF bits.

4.2 Formalization of the Attack Success

In the following, we provide theoretical insights into the success probability that an attacker can achieve by formally modelling the side-channel observations.

The noise $\mathcal{N}_{attack}$ that the attacker is confronted with is a combination of the noise from measurements $\mathcal{N}_{meas.} \sim \mathcal{N}(0, \sigma_{meas.})$ and the inherent noise $\mathcal{N}_{osc.} \sim \mathcal{N}(\mu_{osc.}, \sigma_{osc.})$ of the oscillation frequency f that occurs from measurement to measurement. Assuming that the noise terms are normally distributed and additive, the overall noise is expressed as

$$\mathcal{N}_{attack} = \mathcal{N}_{meas.} + \mathcal{N}_{osc.} \sim \mathcal{N}(\mu_{adv.}, \sigma_{adv.}), \tag{5}$$

where $\sigma_{adv.} = \sqrt{\sigma_{meas.}^2 + \sigma_{osc.}^2}$[3]. Assuming that environmental conditions change slowly, any perturbation is constant among the different frequencies. Thus, offsets $\mu_{osc.}$ of the oscillators are cancelled out when calculating the frequency difference $df = f_C - f_{\neg C}$, and it follows that $\mu_{adv.} = 0$.

The following notation is adopted for the PDF of a normally distributed variable with mean μ and standard deviation σ

$$\phi^\star(x; \mu, \sigma) := \frac{1}{\sigma}\phi\left(\frac{x-\mu}{\sigma}\right) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \tag{6}$$

where $\phi(x)$ is the standard normal distribution.

The attacker mimics the reconstruction process by estimating bounds $\pm T1^\star$, $\pm T2^\star$ and $\pm a^\star$ from the observed values $df^\star$ and guesses $\hat{k}_C$ using Eq. (4). Note, that for $\sigma_{meas.} = 0$ this corresponds to the reconstruction procedure on the device as only $\sigma_{osc.}$ is present compared to the enrollment, i.e., $T1^\star = T1'$, $T2^\star = T2'$ and $a^\star = a'$. In this case, the attacker has the same information as the device and the attack will succeed. We will investigate how the attack success is affected if the attacker observes $\sigma_{meas.} > 0$. Without loss of generality we can set $\sigma_{osc.} = 0$ in the following such that $\sigma_{adv.} = \sigma_{meas.}$ is the additional noise the attacker observes. In other words, the device will reconstruct based on the same bounds as during enrollment, i.e., $T1' = T1$, $T2' = T2$ and $a' = a$,

[3] Note that the device observes only $\sigma_{osc.}$ during reconstruction, i.e., the attacker is always in a worse position compared to the reconstruction.

while the attacker does so based on the noisy versions $T1^\star$, $T2^\star$ and $a^\star$. In the following, we will investigate the success probability

$$Pr_{success}(df_C, \sigma_{adv.}) = Pr[\hat{k}_C = k_C | df_C, \sigma_{adv.}] \tag{7}$$

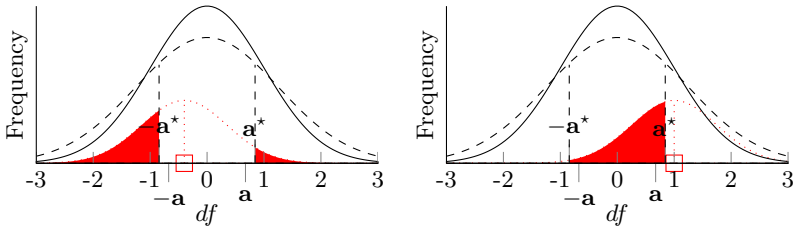
that defines whether an attacker can retrieve the correct PUF bit for a challenge C. The assumption is that the device sees df_C for reconstruction, while the attacker observes $df_C^\star$ drawn from the normal distribution $\phi^\star\left(df^\star; df_C, \sigma_{adv.}\right)$. Besides a relationship of attack success and Signal-to-Noise Ratio (SNR), Eq. (7) also provides insights whether certain values df_C can be more easily attacked.

Weighting the success probability by the occurrence of the df, which follows – per assumption from Section 2 – a normal distribution that for the sake of simplicity we assume to be transformed into a standard normal distribution for $\sigma_{osc.} = 0$, yields the average success probability

$$\overline{Pr}_{success}(\sigma_{adv.}) = \int_{-\infty}^{\infty} Pr_{success}(df, \sigma_{adv.}) \cdot \phi^\star(df; 0, 1)\, \mathrm{d}df. \tag{8}$$

4.3 Exploiting the Two-metric Helper Data Method

In this section, we investigate the side-channel vulnerability of the TMH introduced in Section 2 and provide insights into how attacks with and without helper data knowledge differ. Subsequently, we introduce an extension of the TMH in Section 4.4 that impedes attacks without helper data knowledge.



(a) $|df| < a$ (b) $|df| \geq a$

Fig. 2: Visualization of the attack failure for attacker without helper data.

Attacker without helper data knowledge In a first step we will show that even if an attacker does not know the helper data, i.e., whether metric $M1$ or $M2$ is applied, the TMH leaks side-channel information. From Fig. 1 the attacker needs to observe the magnitude of the frequency difference $df^\star$ regarding $a^\star$: if $|df^\star| > a^\star$, the guessed PUF bit is $\hat{k}_C = 0$, otherwise $\hat{k}_C = 1$ and the success

probability is

$$
\begin{aligned}
Pr^{no\,HD}_{success}(df, \sigma_{adv.}) &= 1 - Pr[\hat{k}_C \neq k_C] \\
&= 1 - \left(Pr\left[|df^\star| \leq a^\star \;\middle|\; |df| > a\right] + Pr\left[|df^\star| > a^\star| \;\middle|\; df| \leq a\right]\right) \\
&= 1 - \begin{cases} \int_{-\infty}^{-a^\star} \phi^\star\left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star + \int_{a^\star}^{\infty} \phi^\star\left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star, & -a \leq df \;\; < a \\ \int_{-a^\star}^{a^\star} \phi^\star\left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star, & -a > df \geq a \end{cases},
\end{aligned} \tag{9}
$$

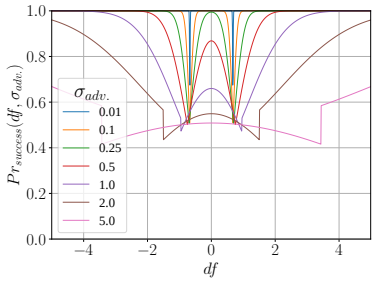
i.e., the probability that the estimated PUF bit $\hat{k}_C$ does not match the correct k_C. Figs. 2a and 2b depict the distributions of $df/df^\star$ from which device and attacker derive their bounds $a/a^\star$as solid and dashed curves respectively. The dotted red curve represents the distribution of observed values $df^\star_C$ for an enrolled value df_C marked as square. Consequently, the filled area below the dotted curve marks a failed attack according to Eq. (9).

Fig. 3a depicts the success probability depending on the enrolled value of df for different levels of noise $\sigma_{adv.}$. Around $df^\star = |a^\star|$, the attacker faces the biggest uncertainty, which is in accordance with intuition as the additional noise changes the retrieved PUF bit most easily close to the decision boundary. Note that the attack does not change whether the *temporal masking* [18] is applied or not – in both cases, the magnitude reveals the PUF bit.

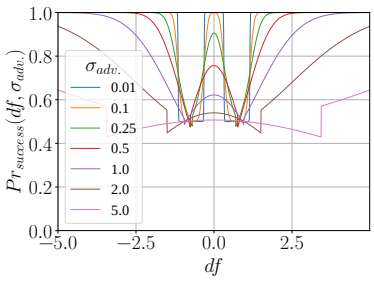
In Fig. 3c the overall success rate for varying noise levels $\sigma_{adv.}$ is depicted according to Eq. (8). In case the helper data is known, but no temporal masking is applied the TMH can be attacked even more easily highlighting that the TMH scheme without further protection enables SCA. Note that without temporal masking the frequency difference df would be revealed independently of the helper data scheme, therefore the notion is rather of theoretical interest and we provide the details in Appendix A. However, the results show that the reliability information of the TMH improves the attack compared to the scenario without helper data knowledge.

Attacker with helper data access and temporal masking Finally, we consider the case where the temporal masking is activated and the attacker cannot trust the sign of the observed frequency difference df_C. The attacker is still able to estimate bounds $\pm T1^\star$ and $\pm T2^\star$, but due to the randomization of the sign there may be a small estimation error. We neglect this effect in the following as it can be considered as an additional noise term in $\sigma_{adv.}$.

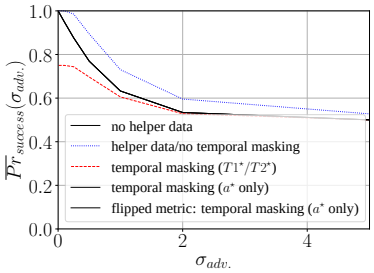
From an attackers point of view there are two possible approaches towards the temporal masking scheme. First, any helper data knowledge can be ignored, i.e., the bound $a^\star$ is used on the absolute values $|df^\star|$ – the attack success rate is the same as if no helper data was known. Second, the attacker can try to exploit the helper data by using the bounds $T1^\star$ and $T2^\star$. However, it has to be considered that the sign of the observation could be flipped. Taking metric $M1$ as an example, the average over the usage of $-T1^\star$ and $T2^\star$(as defined by $M1$



(a) no helper data / temporal masking (using $a^\star$)



(b) temporal masking (using $T1^\star/T2^\star$)



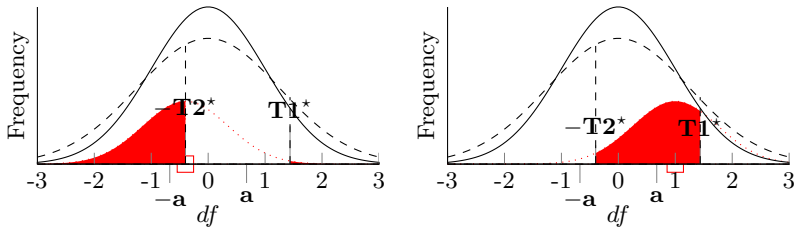
(c) Average success rate

Fig. 3: Simulation of the attack success probability for different levels of attacker noise $\sigma_{adv.}$. (a) – (b) Depeding on enrolled value df. (c) Integrated success probability according to the occurrence of df.

in Fig. 1) and $-T2^\star$ and $T1^\star$(reflecting a sign flip depicted in Fig. 4) gives

$$\begin{aligned} P_1(df, \sigma_{adv.}) &= Pr[\hat{k}_C \neq k_C | w_C = M1, df > a] \\ &= \frac{1}{2}\left[\int_{-T1^\star}^{T2^\star} \phi^\star\left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star + \int_{-T2^\star}^{T1^\star} \phi^\star\left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star\right]. \end{aligned} \tag{10}$$

Accordingly the other intervals from the enrollment are dealt with. From Fig. 4 the resulting error can be seen if the observed $df^\star$ has flipped sign. Fig. 3b highlights that the intervals between $-T1^\star$ and $-T2^\star$ and $T1^\star$ and $T2^\star$ are most prone to errors. For increasing attacker noise $\sigma_{adv.}$ the intervals $[-T1^\star, T2^\star]$ and $[-T2^\star, T1^\star]$ overlap increasingly, i.e., the average error in Fig. 3c converges towards the error for the case without helper data. However, even in the noise-free case and for low-noise scenarios the attack using $T1^\star$ and $T2^\star$ yields worse results compared to only using the helper data or $a^\star$. Thus, with temporal masking activated, the attacker will use the bounds $\pm a^\star$, and no additional information is achieved from the helper data.



(a) $-a < df < 0$ (b) $df \geq a$

Fig. 4: Visualization of the attack failure with known helper data and temporal masking, where the sign of $df^\star$ is flipped compared to df for metric $M1$.

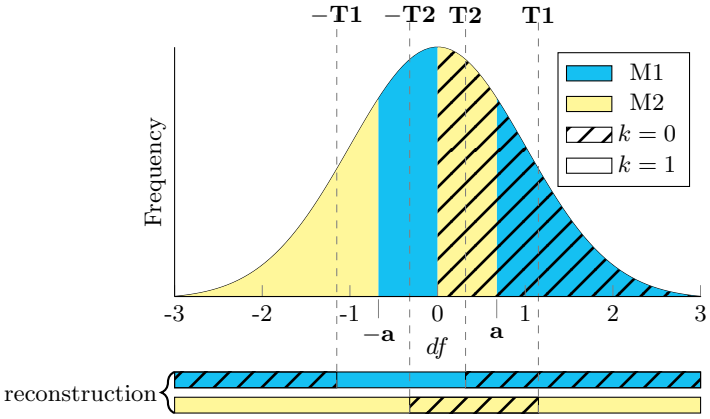


Fig. 5: Choice of metric and extraction of PUF bit value from the the frequency difference df with new metric $M2$.

4.4 SCA-hardening for the Two-metric Helper Data Method

A straightforward improvement of the TMH is to modify the mapping of the metrics. If the bit value with metric M2 is inverted compared to Eq. (2), i.e.,

$$M1 : k = \begin{cases} 0, T2 \leq df \vee df < -T1 \\ 1, -T1 \leq df \ < T2 \end{cases} \qquad M2 : k = \begin{cases} 1, T1 \leq df \vee df < -T2 \\ 0, -T2 \leq df \ < T1 \end{cases}, \tag{11}$$

the PUF response is related to the sign of df as shown in Fig. 5. Note that the choice of the metric, i.e., the helper data, is maintained according to Eq. (3). As the magnitude no longer reveals information the temporal masking protects the TMH scheme as long as the helper data is unknown to the adversary.

However, in case of known helper data, the attacker can still learn about the secret despite the temporal masking countermeasure. As the sign of $df^\star_C$ is randomly altered by the temporal masking, consider the absolute values $|df^\star|$ and estimate the parameters $\pm a^\star$, $\pm T1^\star$ and $\pm T2^\star$ as described in Section 4.2.

Again, the attacker uses the bounds $\pm a^\star$ and combines the helper data and the values of $|df_C^\star|$ to estimate the PUF bit as

$$\hat{k}_C = \begin{cases} 0, & \left(|df_C^\star| > a^\star \ \wedge \ w_C = M1\right) \ \vee \ \left(|df_C^\star| \leq a^\star \ \wedge \ w_C = M2\right) \\ 1, & \left(|df_C^\star| > a^\star \ \wedge \ w_C = M2\right) \ \vee \ \left(|df_C^\star| \leq a^\star \ \wedge \ w_C = M1\right) \end{cases} . \tag{12}$$

Using Eq. (12), the attacker achieves the same success probability as for the non-flipped metric. However, using the sign instead of the magnitude to derive PUF bits, the following improvements are achieved:

1. Attacks without helper data are impeded completely.
2. The Hamming weight of the key is not leaked for unknown helper data, because the attacker cannot distinguish regions that map to 0 or 1.

Yet, the attacker is able to retrieve PUF bits even under noisy measurements. Furthermore, from the knowledge of the likelihood of a correct estimate, a smart guessing strategy can be derived: In Eq. (12), the closer the values of $|df_C^\star|$ is to the boundary $a^\star$, where 0 and 1 change, the lower is the reliability of the estimate. In the remainder of this section we will provide practical results from side-channel measurements to verify the possibility of a successful SCA attack.

4.5 Experimental Setup

We confirm the side-channel evaluation of the TMH scheme using an field programmable gate array (FPGA) implementation of a 63-bit Loop PUF, similar to [18]. The Loop PUF is implemented on an Artix-7 (XC7A100TFTG256) running at f_{clk}=100 MHz. The use of a ChipWhisperer 305 Artix FPGA Target (CW305) SCA board facilitates the analysis as power measurements can be directly acquired using the SMA jack `X4`. We measure the voltage drop of the FPGA's internal supply voltage $\texttt{VCC}_{\texttt{int}}$ over a 100 mΩ shunt amplified by the board's 20 dB low-noise amplifier. A PicoScope 6402D USB oscilloscope acquires measurements at a sampling frequency of f_s=1.25 GHz.

At the beginning of each challenge, a trigger signal allows optimal alignment of the measurements, which are transformed into the frequency domain after acquisition. The counter values after each challenge are read back for offline-verification of the SCA measurements. From the maximum of the frequency signals, the counter values and their differences are estimated. The attack from Section 4.4 is carried out by first performing an enrollment on the actual counter values averaged over ten runs. From the enrollment, a set of reference helper data is generated, which is used throughout the attack. The measurements of challenge pairs $(C, \neg C)$ are randomized to emulate the temporal masking countermeasure.

4.6 Practical attack results

For the following practical results, the TMH is derived on measured frequencies, i.e. the attacker has an additional "noise term" due to the difference of frequencies during enrollment and measurement.

Table 1: Remaining entropy in bit after smart guessing on 63-bit Loop PUF using TMH with flipped metric $M2$ and temporal masking.

campaign	Remaining entropy in bits									
#1	7.3	9.1	15.7	16.6	16.6	17.0	17.5	> 20	> 20	> 20
#2	10.5	11.4	13.1	13.7	14.4	15.6	16.8	19.5	> 20	> 20

Table 1 depicts the results for two different campaigns of ten Loop PUF runs each recorded on the same device. Enrollment is performed on the average of the actual counter values of the campaign, while attacks are carried out on single runs from the measured frequencies. The smart guessing changes first bits derived from frequency difference $|df^{\star}_{C}|$ close to $a^{\star}$ and is stopped at 20 bit to limit the time used for the attack. The median guessing complexity is around 16 bits in both campaigns, indicating that an attack can break the TMH even with the improved construction from Section 4.4 with reasonable effort.

5 Protection of the Two-metric Helper Data Scheme

According to Section 4 an attacker with helper data knowledge and side-channel observations of frequencies can easily break the TMH scheme. However, the matching of frequency differences and helper data is needed. Consequently, the protection is to *hinder the matching*. This is achievable through randomization of the measurement order or, equivalently, the order of applying challenge pairs. Possible solutions are to generate challenge pairs in a randomized order or to store a randomized mapping of helper data and challenges in protected NVM. In a PUF scenario with publicly stored helper data no protected memory shall be used. Further, by Kerckhoffs's principle, the attacker knows how the challenges are generated and applied. Therefore, this section investigates methods to randomly permute challenge pairs during reconstruction.

5.1 Attack Vector on the Protection Mechanism

We consider the case of a flipped metric as introduced in Section 4.4 and temporal masking enabled. Thus, the attacker measures frequency differences df of the Loop PUF without knowing the sign. Even if we randomize the order of the different df, she still knows *which* frequency differences are used. Thus, a divide and conquer strategy is possible: In a first step, the attacker brings the frequencies into the correct order; In a second step, she mounts the attack from Section 4.6 on the ordered data. The attack succeeds if the first step yields only few possible mappings between helper data and frequency differences: For each possible mapping, the attack from Section 4 is to be performed for which the remaining entropy after a single measurement is determined by noise. As a consequence, the difficulty of the attack is mainly determined by the complexity to bring the frequencies into the correct order.

Realistic Loop PUFs generate $M > 3$ secret bits from M challenge pairs $(C, \neg C)$ and their corresponding *df* values. In such a case $M! > 2^M$ different permutations of *df* exist and permutation suffices to protect the secret. The difficulty is to develop an algorithm, which generates the permutations or a subset of permutations efficiently and unpredictably.

5.2 True Random Number Generator Based Protection

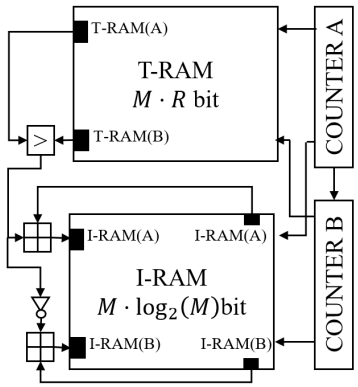
We first development a countermeasure that randomizes the order of challenge pairs $\mathfrak{C}_i = (C_i, \neg C_i)$ using a True Random Number Generator (TRNG). For this purpose at least M random numbers of length $R \geq \lceil log_2(M) \rceil$ bit are needed to index $\mathfrak{C}_i$ and to select the corresponding challenge pair. In this case, an attacker with SCA knowledge observes a random permutation of frequency differences, which hinders matching of helper data with observations and, eventually, the attack in Section 4.

Two questions have to be addressed: (i) What is a suitable choice of R to mitigate SCA attacks while retaining a low implementation overhead and (ii) how to map from random values to a unique sequence of challenge pairs? We discuss the former together with the problem of colliding random numbers in Section 6.1. An efficient method to solve the latter problem is given in algorithmic form and as block diagram in Fig. 6. It requires two $\lceil log_2(M) \rceil$-bit adders, two $\lceil log_2(M) \rceil$-bit counters, an R-bit comparator and $M \cdot (\lceil log_2(M) \rceil + R)$ bit of storage capability. The permutation generator, which is related to Lehmer encoding, takes as an input M distinct randomly chosen R-bit numbers stored in `T-RAM`. The index RAM (`I-RAM`) is initialized with zero. By iterating with counters `A` and `B` over all $\frac{M \cdot (M-1)}{2}$ pairs of random numbers and incrementing always the entry in `I-RAM` that corresponds to the index of the larger random number, the `I-RAM` finally contains a permutation of the values $0, ..., M-1$. The sorting algorithm defines – independent of the number of random bits R – the permuted order of $\mathfrak{C}_1$ to $\mathfrak{C}_M$. Please note, that the order is unpredictable for the attacker but can be resolved by the device in order to match frequency differences and helper data.

The TRNG-based approach described so far has one significant drawback: It requires a relatively large number of random bits. This raises the question, if a more lightweight protection mechanism is possible.

5.3 Towards a Lightweight Protection of the TMH Method

The TRNG-based approach from Section 5.2 exclusively focuses on the complexity of guessing the sorting of the frequency differences. However, in a practical setting, frequency differences as well as their relations to each other are not constant between multiple reconstructions of the PUF response due to PUF noise. This limits the attacker's capability to establish a relationship between observed frequency differences from different reconstructions. In parallel, the measurement complexity can limit the applicability of the attack. This leads to the question if a more lightweight approach with less random bits compared to Section 5.2, and lower implementation complexity is feasible that retains a



Algorithm 1: Permutation Generator

Choose M R-bit distinct random numbers r_i
Store r_i to `T-RAM`.
for `A` in 0 to $M-1$ **do**
  for `B` in `A`+1 to M **do**
    if `T-RAM(A)` > `T-RAM(B)` **then**
      increment `I-RAM(A)`
    else
      increment `I-RAM(B)`
    end if
  end for
end for
return Sequence in `I-RAM`.

Fig. 6: Block diagram and algorithm of permutation generator

sufficient level of protection. This section introduces such an approach with the implementation shown in Fig. 7. It uses a combination of the linear feedback shift register (LFSR) seed (b_0 to b_5), the LFSR feedback polynomial (in red), an additive mask (in green), and clock shifting (in blue). We discuss an LFSR

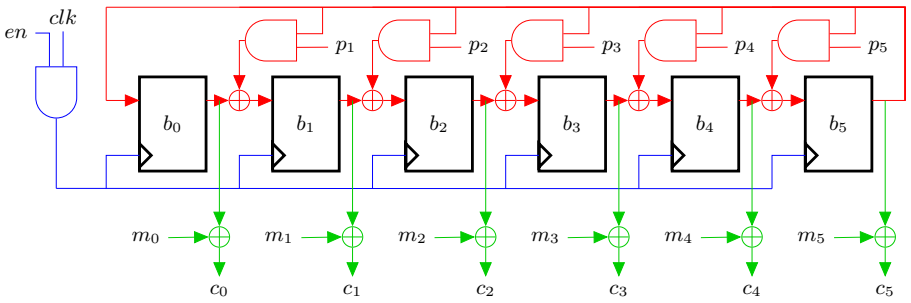


Fig. 7: Combined low-complexity countermeasure.

with six state bits b_0 to b_5, because our target application is a Loop PUF with 64 stages. Hence, 63 challenges[4] have to be permuted, and the $2^N - 1$ states of an $N = 6$-bit LFSR directly represent the index of a challenge, which makes the approach very lightweight. The discussion focuses on a 6-bit LFSR as the Loop PUF with more than 64 stages would be too slow and less stages ease the attack. For other applications longer LFSR length could be considered, i.e., the method is generally applicable.

[4] From the 64 Hadamard challenges the pair of the all-zero and the all-one challenge shall not be used to derive secret bits, c.f. [2, 18] for further details.

Entropy from the random seed The first randomization technique under investigation is the use of a random initialization of the LFSR state bits. Since all states of the LFSR are used to index the $2^N - 1$ challenge pairs, the random seed corresponds to a cyclic shift of the LFSR output. An attacker has to guess the correct seed to obtain the correct sorting of the frequency differences, which introduces $log_2(2^N - 1) \approx N$ bit of entropy – 5.98 bit in case of the 6-bit LFSR.

Entropy from the random shifts The second protection mechanism is a multi-bit shift of the LFSR output. The LFSR is read out only after every n-th shift, where n is selected once per reconstruction. In Fig. 7 the enable signal en is set for n cycles after which the values of the c_i are used to determine challenge index. As the LFSR output is cyclic the shift by n has to be relative prime to the period of the LFSR $2^N - 1$ in order to reach all states of the LFSR, i.e., to index all challenge pairs. In case of the 6-bit LFSR, from Euler's totient function $\varphi(63)$ there are 36 values for n corresponding to approximately 5.12 bits of guessing entropy for the attacker.

Note that shifting the LFSR output has two possible drawbacks: First, the method delays the indexing of the challenge pairs. This is, however, not critical, since the next index is calculated in parallel to the much slower measurement of the Loop PUF. Second, an attacker can observe through a side-channel how frequently the LFSR is clocked. However, the SNR for the attacker is likely too small, to observe the LFSR: When the LFSR is clocked in parallel to the Loop PUF, the attacker has to observe both the Loop PUF frequency and the LFSR in parallel and the attacker can observe the clocking only $2^N - 1$ times, i.e., once per frequency difference, since for the next reconstruction another n is chosen. An additional hiding countermeasure could be to randomize the point in time, when the LFSR starts shifting during the measurement time of the Loop PUF.

Entropy from the random mask The third method to add randomness is to mask the output by applying an additive 6-bit mask to the LFSR state corresponding to 6 bits of entropy. By adding a mask we introduce the index zero, i.e., the index of a challenge pair, which is not used, while another index – the one equal to the mask – disappears. We therefore map the zero index to the missing index in our analysis[5].

Entropy from the random polynomials As a fourth and last method different feedback polynomials are randomly chosen for the LFSR. For the case of $N = 6$ there are six irreducible polynomials [8] that add another $\log_2(6) \approx 2.6$ bit of entropy.

[5] We also investigated inserting a zero randomly at the beginning or end of the state before masking. For index zero we selected the Loop PUF under the all zero/all one challenge. However, the results were equivalent to the ones shown in this work.

Summary of Countermeasure The idea of the combined countermeasure is to increase the complexity for an attack while maintaining a low complexity of the countermeasure. Summing the entropy values of the four protection mechanisms, 19.7 bit of entropy are achieved for an 6-bit LFSR. While the brute-force complexity below 20 bit would not prevent an attack, recall that the attacker has to perform the smart guessing attack from Section 4.6 up to $2^{19.7}$ times if the different sequences are indistinguishable. Considering that a single run of the attack takes more than an hour, even with a 100-fold parallelization, the attack on the different sequences would take approximately one year, which can be considered a reasonable protection level for a lightweight solution. However, the practical security analysis in Section 6 shows that an adversary can distinguish sequences generated by the lightweight countermeasure, which reduces the entropy and allows for an attack.

6 Security Analysis

The attacker can observe absolute values of frequency differences, but due to temporal masking the sign of the differences is unknown. The goal is to enable the attack from Section 4 by reconstructing the order of the frequency differences. Since the adversary does not have direct information regarding the correct ordering, she can enable an attack by labeling the observed frequencies with symbols. Then she brings the symbols into an order that might have been generated by the protection mechanism. An attacker wins if she can guess or identify the correct ordering of the frequencies since then and only then she can sort the frequency differences according to the helper data.

6.1 Security Analysis of the TRNG-Based Protection

If a TRNG outputs only distinct random numbers, the protection mechanism in Section 5.2 does not reveal information about the sorting. However, in practice collisions, i.e., sampling the same random number twice, are possible. Options to overcome this problems include to put the frequency differences for which collisions appear into a predefined order or to re-sample in case of a collision. The former leads to a higher probability for specific permutations that gives additional information to an attacker. To prevent possible statistical attacks exploiting permutations with distinct probabilities, we suggest to ensure distinct random numbers through re-sampling.

The probability that collisions of at least two random numbers appear is defined by the Birthday Paradox. However, it is less important if collisions appear than how many bits are required when generating a set of distinct random numbers if we resolve the collisions through re-sampling. Let us assume that the random numbers r_i are sampled sequentially and that the current r_i is re-sampled until the TRNG provides a number not yet used. Further, let us assume an ideal TRNG providing R-bit outputs such that all 2^R possible sequences are equally likely. Under this assumptions the probability of a collision $p_{re,R}(i)$ for

the i-th random number with $2^R \geq i \geq 0$ is linked by

$$p_{re,R}(i) = \frac{i}{2^R} \Leftrightarrow E_{re,R}(i) = \frac{1}{1 - \frac{i}{2^R}} = \frac{2^R}{2^R - i}$$

with the expected number of samples to receive a yet unused random number $E_{re,R}(i)$. As a consequence, the average number of required random bits $E_{bits,M}(R)$ when sampling all random numbers $r_i,\ i \in \{0, ..., M-1\}$ is

$$E_{bits,M}(R) = R \cdot \sum_{i=0}^{M-1} \frac{2^R}{2^R - i}, \tag{13}$$

which has a minimum in R. For a given M the minimum defines the optimal choice of R regarding the expected amount of TRNG bits needed. We suggest to use this optimum for the permutation generator from Section 5.2: Under consideration of re-sampling for the case of $M = 63$ the minimum is reached at $R = 8$, i.e., on average $E_{bits,63}(8) \approx 577.23$ bits are needed.

6.2 Security Analysis of the Lightweight LFSR

In this section, we analyze the quality of the lightweight countermeasure from Section 5.3 in order to show its limitations and to point towards possible solutions. We discuss the different countermeasures individually and show practical evaluation results of individual and combined countermeasures. In the following, we interpret the Galois LFSR state from Fig. 7 as integers, e.g., $[1, 0, 0, 0, 0, 0]$ corresponds to 1. For simplicity of explanation we treat the case of a 6-bit LFSR.

Attack Strategy Assume an attacker taking two measurements from two distinct reconstructions of the same Loop PUF. The randomized seed, shift size, mask, and polynomial are fixed for each reconstruction. The attacker defines the frequency differences of the PUF as symbols $s_i \in 1, ..., N$. She knows that there is a native order $\mathbf{s}_{nat} = [s_1, ..., s_N]$ of the symbols, which matches the sorting of the helper data. Further, the frequency differences $\mathbf{s}_{obs}$ she observes are sorted by a permutation described by a permutation matrix $\mathbf{A}$.

From the $2^{19.7}$ bit of entropy Section 5.3, more than $850k$ permutation matrices exist and the attacker's task is to find the correct one corresponding to one of her observations. If one or more randomization options are disabled, the number of permutations decreases accordingly. The attacker uses a differential approach on the observed sequences $\mathbf{s}_{obs,1}$ and $\mathbf{s}_{obs,2}$. She resorts each of the sequences with all possible sub-sequences. For the correct permutation matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ of two noise-free measurements it holds that

$$(\mathbf{s}_{obs,1} = \mathbf{s}_{nat}\mathbf{A}_1 \wedge \mathbf{s}_{obs,2} = \mathbf{s}_{nat}\mathbf{A}_2) \Rightarrow \mathbf{s}_{nat} = \mathbf{s}_{obs,1}\mathbf{A}_1^{-1} = \mathbf{s}_{obs,2}\mathbf{A}_2^{-1}.$$

However, the reverse argumentation does *not* hold, i.e., if two matrices $\mathbf{A}_1^\star$ and $\mathbf{A}_2^\star$ exist such that $\mathbf{s}_{cand} = \mathbf{s}_{obs,1}\mathbf{A}_1^{\star -1} = \mathbf{s}_{obs,2}\mathbf{A}_2^{\star -1}$ the candidate solution $\mathbf{s}_{cand}$

is not necessarily $\mathbf{s}_{nat}$. In the following we determine how many solutions $\mathbf{s}_{cand}$ exist, i.e., how much entropy the attacker faces.

For noisy measurements, a direct matching of the resorted sequences is not reasonable. However, we show that the attack is still applicable by correlating the two observed and transformed sequences $\mathbf{s}_{obs,1}\mathbf{A}_1^{\star -1}$ and $\mathbf{s}_{obs,2}\mathbf{A}_2^{\star -1}$.

Confusion from random seed Randomizing the seed corresponds to a cyclic shift of the LFSR. The permutation matrix $\mathbf{R}_m$ for a cyclic shift by m bit is $\mathbf{R}_m = \mathbf{R}_1^m$, where $\mathbf{R}_1$ is the permutation matrix of the shift by one bit. Same applies to the inverse, i.e., $\mathbf{R}_m^{-1} = \left(\mathbf{R}_1^{-1}\right)^m$. Consequently, if two observations have the shifts α and β from their seeds, the relative shift of the sequences corresponds to a κ-bit shift with $\kappa = \alpha - \beta$. The native sequence follows from inverting the respective shifts, i.e., $\mathbf{s}_{obs,\alpha}\mathbf{R}_\alpha^{-1}$ and $\mathbf{s}_{obs,\beta}\mathbf{R}_\beta^{-1}$. Every candidate $\mathbf{s}_{cand,n}$ that fulfills

$$\mathbf{s}_{cand,n} = \mathbf{s}_{obs,\alpha}\mathbf{R}_\alpha^{-1}\mathbf{R}_n = \mathbf{s}_{obs,\beta}\mathbf{R}_\beta^{-1}\mathbf{R}_n = \mathbf{s}_{nat}\mathbf{R}_n$$

is a solution. Since the LFSR is cyclic with period $2^N - 1$, the attacker cannot distinguish $2^N - 1$ different sequences.

Confusion from random shift Similarly to the previous argumentation, the shift by multiple bits is a permutation with a permutation matrix $\mathbf{T}$. Let $\mathbf{T}_\alpha$ corresponds to the permutation of the LFSR state sequence under shifts by α, and $\mathbf{T}_\beta$ corresponds to the permutation of the LFSR state sequence under shifts by β. Since all shift sizes are relative prime to the LFSR length $2^N - 1$, their product modulo $2^N - 1$ is relative prime to the LFSR length. The multiplication of matrices $\mathbf{T}_\alpha$ and $\mathbf{T}_\beta$ therefore results in a valid shift. Thus, for each pair of observations there exists a pair of matrices $\mathbf{T}_k$, $\mathbf{T}_l$ so that

$$\mathbf{s}_{obs,i}\mathbf{T}_\alpha^{-1}\mathbf{T}_k = \mathbf{s}_{obs,i}\mathbf{T}_\beta^{-1}\mathbf{T}_l,$$

and the attacker cannot distinguish different shift widths.

Confusion from random masks A mask is implemented as a bitwise XOR onto the LFSR state with the all-zero result mapped to the mask value. Different from the previous methods, two permutations M_α and M_β inserted by the mask are unique. As a consequence, if different masks are used to permute the state sequence of the LFSR, the resulting symbol orders can be distinguished since only for the correct pair of masks and – as the experiments show – few exceptions

$$\mathbf{s}_{obs,\alpha}\mathbf{M}_\alpha^{-1} = \mathbf{s}_{obs,\beta}\mathbf{M}_\beta^{-1}$$

holds. Consequently, the entropy spent through this countermeasure does *not* contribute to the confusion of the attacker. In addition, the experiments in the last part of this section reveal that the mask, when combined with the random shift, effectively reduces the uncertainty for an attacker.

Confusion from random polynomial Similar to the mask case, polynomials do not lead to an increased confusion of the attacker but rather allow for a better attackability of the LFSR. The reason is, that the permutation matrices P_α and P_β from different feedback polynomials are very distinct. Therefore,

$$\mathbf{s}_{obs,\alpha}\mathbf{P}_\alpha^{-1} = \mathbf{s}_{obs,\beta}\mathbf{P}_\beta^{-1}$$

only holds for the correct permutation and – as the experiments show – for few exceptions.

Practical evaluation We verify the theoretical insights with experimental data from synthetic symbols as well as on the measurement data of campaign #1 used in Table 1. All experiments assume temporal masking, i.e., the absolute values of frequency differences are used. An attack is limited to the measurements of a single Loop PUF, but can employ measurements from multiple reconstructions. Each pair of two reconstructions in the campaign is analyzed. For the attack, resorted frequency differences from two reconstructions are correlated in order to find their correct ordering. A correlation of 1 would be a perfect match of sequences, which only occurs for synthetic data. As frequency differences differ from measurement to measurement, for experimental data the correlations depend on the noise level. For fair comparison we present results for highest and lowest correlation, i.e., for lowest and highest noise seen by the attacker. This best and worst case have correlations of 0.97 and 0.91 for our measurements. Please note, that the attack requires only two single-shot measurements of the Loop PUF.

The synthetic symbols and measured frequency differences are permuted in software with different permutation strategies from Section 5.3 enabled. In accordance with the attack strategy, the attacker pre-computes all inversions to map from some permutation back to the native sorting. For the 6-bit LFSR a list with more than $850k$ inversions is generated. Then, the attacker permutes the two observed sequences of symbols w.r.t. to the pre-computed inversion list and correlates the result. Clearly, the result between the two correct inversions is $\rho_0 = 1$ in case of a noise free sequence and $\rho_0 = 0.91$ and $\rho_0 = 0.97$ in case of the selected noisy sequences. The complete attack takes on a commodity computer[6] in the range of seconds if only one permutation strategy is enabled up to less than 70 minutes with all four protection mechanisms enabled.

Table 2 summarizes the results for different levels of countermeasures enabled, namely random seed (**R**), randomly selected shift (**T**), random mask (**M**), and randomly selected feedback (**P**). Each experiment is repeated ten times for each set of enabled countermeasures and minimum, maximum, and median number of indistinguishable sequences are provided. A sequence is included into the set of possible candidates if it yields a correlation $\rho \geq \rho_0 - \varepsilon$ with the data. The correlation threshold ρ_0 considers the noise level of the data sets and setting $\varepsilon = 10^{-6}$ prevents rounding errors. The values indicate, how many times the

[6] Intel(R) Core(TM) i7-6700 CPU; 3.40GHz; 4 cores; 16GB RAM

Table 2: Attack results on lightweight protection mechanism. Noisy data from power measurements of Loop PUF frequencies, noise free data from synthetically generated symbols. **R**, **T**, **M**, **P** correspond to random seed, randomly selected shift width, random mask, and randomly selected polynomial.

Enabled Countermeasure				Used Data								
				Noise free			Noisy; correlation 0.97			Noisy; correlation 0.91		
R	**T**	**M**	**P**	min	median	max	min	median	max	min	median	max
x	-	-	-	63	63	63	63	63	63	63	63	63
-	x	-	-	36	36	36	36	36	36	36	36	36
-	-	x	-	1	1	1	1	2	8	1	8	27
-	-	-	x	1	1	1	1	1	6	1	1	1
x	x	-	-	2,268	2,268	2,268	2,268	2,268	2,268	2,268	2,268	2,268
x	x	x	-	378	378	756	378	378	3,780	378	2,268	10,584
x	x	-	x	2	2	13,608	2	4	13,608	2	4	13,608
x	x	x	x	2	4	2,268	2	4	11,340	4	13	870,912

attacker would have to run the attack on the TMH scheme in Section 4.4 under different mappings between helper data and frequency differences.

We provide some remarks regarding the results in Table 2:

1. Except for all countermeasures enabled, the minimum value is the same for the noise conditions, and minimum and median are close. The small deviations indicate that the attack is quite robust against noise.
2. The maximum for noisy data and only polynomials (**P**) enabled is 6 and the maximum value for random seed, shift width, and feedback enabled ($\mathbf{R}, \mathbf{T}, \mathbf{P}$) is always $13,608 = 36 \cdot 63 \cdot 6$, both corresponding to the theoretical maximum according to Section 5.3. The reason for these cases is, that by random chance twice the same polynomial has been selected and the attacker does not know which one. Conversely, if distinct polynomials are used, the distinction of two sequences is easier, which suggest that the polynomial countermeasure should not be used in combination.
3. In case that the mask is enabled, the median and maximum numbers of indistinguishable sequences increase, and at the same time the minimum number decreases, compared to the same setting without mask, i.e., ($\mathbf{R}, \mathbf{T}, \mathbf{P}$) vs. ($\mathbf{R}, \mathbf{T}, \mathbf{M}, \mathbf{P}$), and ($\mathbf{R}, \mathbf{T}$) vs. ($\mathbf{R}, \mathbf{T}, \mathbf{M}$). While the increased median and maximum values indicate a susceptibility of the attack towards noise, the increase of the minimum value reveals that masking is an unsuited permutation strategy and lowers the overall protection similar as the use of random polynomials.

Summarizing, the best combination of protection mechanisms is the use of random seeds (**R**) and randomly selected shifts (**T**) for which the attacker faces $63 \cdot 36 = 2,268$ indistinguishable sequences when observing ten different pairs of Loop PUF measurements. While we showed that the attack is independent of the noise level, an attacker could combine N_{meas} measurements to construct $N_{meas}!$ different pairs for an attack. From each pair, processed in parallel, she could take

the $2,268$ most likely results or drop results, which have more than $2,268$ equally high correlations. The resulting up to $N_{meas}! \times 2,268$, sequences could be used in parallel to match helper data and frequency differences and to run the attack from Section 4.6. This eventually demonstrates that the lightweight countermeasure hardly provides sufficient protection to the TMH method. Nevertheless, the discussion highlights pitfalls, e.g., regarding combined permutations, and provides indicators on how to develop improved lightweight protection mechanisms in the future.

7 Conclusion

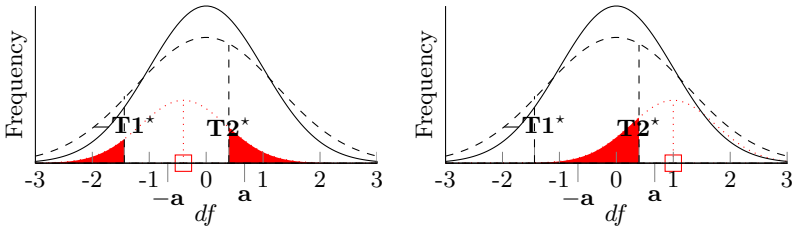
This paper studies the security of a PUF using the TMH method in the presence of SCA attacks. While TMH can greatly enhance the reliability without resorting to ECC, we show that the used metrics need to be modified in order to achieve a high level of security when the helper data is unknown to the attacker. In case of public helper data, it appears that the TMH method has important security weaknesses. Two protections are proposed relying on randomization of the challenge order. The first one, which takes advantage of a TRNG, provides excellent security but requires a significant number of random bits. The second and less costly solution is relying on an LFSR, but only adds a limited security enhancement. The limitations of the approaches highlight the need for more efficient protections in terms of complexity and security. In particular, we are interested in minimizing the number of random bits and in interleaving challenges during oscillation measurements.

References

1. Becker, G.T.: Robust fuzzy extractors and helper data manipulation attacks revisited: Theory versus practice. IEEE Transactions on Dependable and Secure Computing **16**(5), 783–795 (2019). https://doi.org/10.1109/TDSC.2017.2762675
2. Cherif, Z., Danger, J., 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
3. 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. IEEE (2019)
4. Delvaux, J., Verbauwhede, I.: Attacking PUF-based pattern matching key generators via helper data manipulation. In: Benaloh, J. (ed.) Topics in Cryptology – CT-RSA 2014, pp. 106–131. No. 8366 in Lecture Notes in Computer Science, Springer International Publishing (2014), DOI: 10.1007/978-3-319-04852-9_6
5. Delvaux, J., Verbauwhede, I.: Key-recovery attacks on various RO PUF constructions via helper data manipulation. In: 2014 Design, Automation Test in Europe Conference Exhibition (DATE). pp. 1–6 (2014). https://doi.org/10.7873/DATE.2014.085
6. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: EUROCRYPT. pp. 523–540 (2004)

7. Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: FPGA intrinsic PUFs and their use for IP protection. In: CHES. pp. 63–80. Lecture Notes in Computer Science, Springer (2007)
8. Houghton, A.: Error Coding for Engineers. Springer (2001)
9. Katzenbeisser, S., Kocabaş, Ü., Rožić, V., Sadeghi, A.R., Verbauwhede, I., Wachsmann, C.: PUFs: Myth, fact or busted? a security evaluation of physically unclonable functions (PUFs) cast in silicon. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 283–301. Springer (2012)
10. Maes, R., Van Herrewege, A., Verbauwhede, I.: PUFKY: A fully functional PUF-based cryptographic key generator. In: Proceedings of the 14th International Conference on Cryptographic Hardware and Embedded Systems. pp. 302–319. CHES'12, Springer-Verlag, Berlin, Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-33027-8_18
11. Merli, D., Heyszl, J., Heinz, B., Schuster, D., Stumpf, F., Sigl, G.: Localized electromagnetic analysis of RO PUFs. In: 2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST). pp. 19–24 (June 2013). https://doi.org/https://doi.org/10.1109/HST.2013.6581559
12. Merli, D., Schuster, D., Stumpf, F., Sigl, G.: Semi-invasive em attack on FPGA RO PUFs and countermeasures. In: 6th Workshop on Embedded Systems Security (WESS'2011). ACM (Mar 2011). https://doi.org/https://doi.org/10.1145/2072274.2072276
13. Merli, D., Stumpf, F., Sigl, G.: Protecting PUF error correction by codeword masking. IACR Cryptology ePrint Archive **334** (2013), http://eprint.iacr.org/2013/334
14. Pehl, M., Hiller, M., Sigl, G.: Secret Key Generation for Physical Unclonable Functions, p. 362–389. Cambridge University Press (2017). https://doi.org/10.1017/9781316450840.014
15. Schaub, A., Danger, J., Guilley, S., Rioul, O.: An improved analysis of reliability and entropy for delay PUFs. In: 21st Euromicro Conference on Digital System Design, DSD 2018, Prague, Czech Republic, August 29-31, 2018. pp. 553–560 (2018). https://doi.org/10.1109/DSD.2018.00096
16. Shiozaki, M., Fujino, T.: Simple electromagnetic analysis attacks based on geometric leak on an ASIC implementation of ring-oscillator PUF. In: Proceedings of the 3rd ACM Workshop on Attacks and Solutions in Hardware Security Workshop. pp. 13–21. ASHES'19, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3338508.3359569
17. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: Proceedings of the Design Automation Conference, (DAC '07). 44th ACM/IEEE. pp. 9–14 (2007)
18. Tebelmann, L., Danger, J.L., Pehl, M.: Self-secured PUF: Protecting the loop PUF by masking. In: Bertoni, G.M., Regazzoni, F. (eds.) Constructive Side-Channel Analysis and Secure Design. pp. 293–314. Springer International Publishing (2020). https://doi.org/https://doi.org/10.1007/978-3-030-68773-1_14
19. Tebelmann, L., Pehl, M., Sigl, G.: EM side-channel analysis of BCH-based error correction for PUF-based key generation. In: Proceedings of the 2017 Workshop on Attacks and Solutions in Hardware Security. pp. 43–52. ASHES '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3139324.3139328, http://doi.acm.org/10.1145/3139324.3139328

A Attacker with helper data access and no temporal masking



(a) $-a < df < 0$ (b) $df \geq a$

Fig. 8: Visualization of the attack failure for attacker with helper data knowledge. As an example metric $M1$ is used, but no temporal masking is effective.

We assume that the attacker can read the helper data, but the temporal masking countermeasure is not activated. We show how this additional information affects the attack highlighting that the TMH scheme without further protection enables SCA. This notion is rather of theoretical interest as without temporal masking, the frequency difference df would be revealed independently of the helper data scheme. However, the results show that the reliability information of the TMH can also be exploited by the attacker and improves the attack compared to the scenario without helper data knowledge.

Figs. 8a and 8b depict the attack scenario assuming helper data knowledge. As an example, the use of metric $M1$ is depicted, where an attacker can use the bounds $-T1^\star$ and $T2^\star$instead of $\pm a^\star$ if no helper data is known. Compared to Figs. 2a and 2b, the red area below the distribution of observed values is significantly smaller. This indicates that the attacker benefits from the reliability information encoded in the helper data and is formalized in the following.

Assuming metric $M1$ and the value $df > a$ during enrollment the actual PUF bit is k_C=0 according to Eqs. (2) and (3). The attacker will know that $M1$ is the metric but any observed value $T1^\star \leq df'_C < T2^\star$ is decoded as $\hat{k}_C = 1 \neq k_C$. In other words any perturbation $T1^\star - df < \epsilon < T2^\star - df$ will lead to an error in the attack. Now for $df^\star \sim \mathcal{N}(df, \sigma_{adv.})$, the probability for this event is

$$\begin{aligned} P_1(df, \sigma_{adv.}) &= Pr[\hat{k}_C \neq k_C | w_C = M1, df > a] \\ &= \int_{-T1^\star}^{T2^\star} \phi^\star\left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star. \end{aligned} \tag{14}$$

The boundaries $-T1^\star$ and $T2^\star$ depend on the noise the attacker faces[7], thus Eq. (14) establishes a relationship between the SNR and failure probability. Similarly, for the case when the metric is $M1$ and $k_C = 1$, the failure probability is:

$$\begin{aligned} P_2(df, \sigma_{adv.}) &= Pr[\hat{k}_C \neq k_C | w_C = M1, -a \leq df \leq 0] \\ &= \int_{-\infty}^{-T1^\star} \phi^\star\left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star + \int_{T2^\star}^{\infty} \phi^\star\left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star. \end{aligned} \tag{15}$$

In an analogous way the failure probability for metric $M2$ with $k_C = 0$ is defined as

$$\begin{aligned} P_3(df, \sigma_{adv.}) &= Pr[\hat{k}_C \neq k_C | w_C = M2, df < -a] \\ &= \int_{-T2^\star}^{T1^\star} \phi^\star\left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star, \end{aligned} \tag{16}$$

and for metric $M2$ with $k_C = 1$ it results in

$$\begin{aligned} P_4(df, \sigma_{adv.}) &= Pr[\hat{k}_C \neq k_C | w_C = M2, 0 < df \leq a] \\ &= \int_{-\infty}^{-T2^\star} \phi^\star\left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star + \int_{T1^\star}^{\infty} \phi^\star\left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star. \end{aligned} \tag{17}$$

From the probabilities in Eqs. (14) to (17), which define the entire support of df, the overall success probability to recover a PUF bit is given by

$$Pr_{success}(df, \sigma_{adv.}) = 1 - \sum_{i=1}^{4} P_i(df, \sigma_{adv.}). \tag{18}$$

Fig. 9 depicts the success probability for different levels of noise $\sigma_{adv.}$ an attacker faces and depending on the enrollment value df. The results show that $df \approx \pm a$ and $df \approx 0$ contain most uncertainty for the attacker, i.e., it is most likely that the estimated value for the PUF bit k'_C is wrong. The attacker faces the highest uncertainty for values of df close to the boundary between $\hat{k} = 0$ and $\hat{k} = 1$. On the one hand, this means the attack will not yield a 100% success rate for all PUF bits. On the other hand, the attacker is provided with reliability information for the attack results that allow for developing a smart guessing strategy.

[7] Note: For the standard normal distribution $\mu = 0$, $\sigma = 1$, the resulting value are $|\pm T1| = 0.31863936$, $|\pm a| = 0.67448975$ and $|\pm T2| = 1.15034938$. Depending on σ, the value are scaled accordingly. Notably the points that define the octiles are not equidistant.

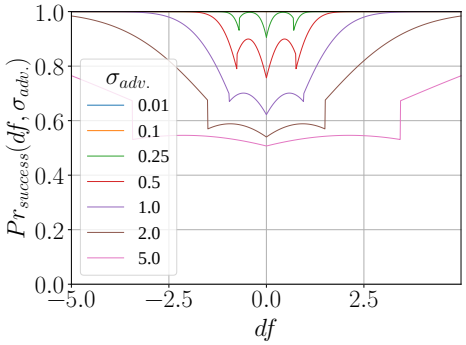


Fig. 9: Helper data/no temporal masking: Simulation of the attack success probability for different levels of attacker noise $\sigma_{adv.}$.