TUM School of Computation, Information and Technology
Technische Universität München

# Side-Channel Analysis and Countermeasures for Physical Unclonable Functions

## Ring-Based PUF Primitives, Bit Derivation and BCH Codes

**Lars Tebelmann**

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

**Doktors der Ingenieurwissenschaften (Dr.-Ing.)**

genehmigten Dissertation.

**Vorsitz:**
Prof. Dr.-Ing. Ulf Schlichtmann

**Prüfer der Dissertation:**
1. Prof. Dr.-Ing. Georg Sigl
2. Prof. Jean-Luc Danger

Die Dissertation wurde am 13.01.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 18.07.2023 angenommen.

# Abstract

This thesis explores Side-Channel Analysis (SCA) of ring-based Physical Unclonable Functions (PUFs), and BCH codes used to correct errors in the PUF response. SCA of the TERO PUF and Loop PUF is conducted. Countermeasures for the Loop PUF are proposed that protect sign- and amplitude-based bit derivation, and can partially be transferred to other PUFs. Differential SCA of BCH codes is extended and the possibility, and limitations of horizontal SCA are analyzed.

# Zusammenfassung

Diese Arbeit untersucht die Seitenkanalanalyse (SCA) von Ring-basierten Physical Unclonable Functions (PUFs) und dabei zur Fehlerkorrektur verwendeter BCH Codes. Eine SCA von der TERO PUF und Loop PUF wird durchgeführt. Gegenmaßnahmen für die Loop PUF werden vorgestellt, die Bit- und Amplituden-basierte Bitableitung schützen und teilweise auf andere PUFs übertragen werden können. Differentielle SCA von BCH Codes wird erweitert und die Möglichkeit und Beschränkungen von horizontaler SCA werden analysiert.

# Acknowledgment

First of all, I would like to thank Prof. Dr.-Ing. Georg Sigl for providing me with the opportunity to pursue the Ph.D. in a supportive environment with the necessary freedom to follow my own research interests. Further, I would like to extend my sincere thanks to Prof. Jean-Luc Danger for co-supervising this thesis. I am extremely grateful to Dr.-Ing. Michael Pehl for the many fruitful discussions that enriched me both, technically and personally, and for providing me with his profound knowledge in the domain of PUFs and hardware design.

Many thanks to the colleagues from the Chair of Security in Information Technology and the Fraunhofer AISEC with whom I had the possibility to collaborate and discuss, and who made these almost six years a very enjoyable period of my life. In particular, I would like to express my gratitude to Thomas Schamberger and Michael Gruber for the joint efforts to set up the hardware security lab, the many discussions about attacks and countermeasures, and five memorable years in the office. Special thanks to Manuel Brosch, who joint me in the quest of horizontal side-channel analysis through all ups and downs. Many thanks to Dr.-Ing. Michael Tempelmeier for technical and non-technical exchange, and providing motivation and diversion where needed.

I am also very thankful to all my co-authors, and people who provided feedback, inspiration and their knowledge to my research: Dr.-Ing. Julian Brosda, Dr.-Ing. Fabrizio De Santis, Maximilian Egger, Robert Hesselbarth, Dr.-Ing. Johann Heyszl, Manuel Ilg, Dr.-Ing. Vincent Immler, Patrick Karl, Dr.-Ing. Ulrich Kühne, Alexander Kulow, Florian Lippert, Matthias Probst, Marc Schink, Emanuele Strieder, Dr.-Ing. Florian Unterstein, and Moritz Wettermann.

I would also like to thank the students I had the opportunity to supervise during all the years. Special thanks to Valentin Huber and Moritz Wettermann who enabled parts of my research through their dedicated work.

Finally, I would like to thank my family and friends for their love, encouragement and joy provided. Words cannot express my gratitude to Lisa whose enduring support and unconditional love has been indispensable for the success of this Ph.D. project.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

In an increasingly interconnected world, the ubiquity of embedded devices leads to an elevated requirement of secure communication and data protection. Cryptographic algorithms are leveraged to achieve security goals such as *authenticity*, *confidentiality* and *integrity*. A common ground of all algorithms is the use of secret keys, e.g., for encryption, decryption and signing messages. However, the proper storage of the key material is a major challenge for embedded devices, where an adversary has potentially physical access to the devices. In particular if the key is kept in Non-Volatile Memory (NVM), attacks during the power-off phase of a device have to be considered. Unprotected storage is prone to probing attacks that allow for reading out the key material, e.g., by optical analysis upon delayering the Integrated Circuit (IC) or related attacks. Therefore, secured NVM is a requirement to maintain system security, which however comes at an increased cost.

Especially low-cost embedded devices are physically accessible and may serve as an entry point for attacks. While such devices require decent security mechanisms, their low-cost nature limits the acceptable cost overhead. One major issue is the secure key storage to provide credentials for e.g., secure firmware updates or authenticated communication. However, secured NVM is frequently not affordable. Also, NVM protection mechanisms require permanent power, draining the limited energy resources of embedded devices in the field.

To overcome the limitations of non-volatile storage, Physical Unclonable Functions (PUFs) provide means for secure key storage without requiring expensive secured NVM. By leveraging manufacturing variations for each device a unique, but reproducible PUF response is generated that can be used to embed a secret key. Compared to classical key storage, the secret is only derived on-demand from the PUF and stored in volatile memory, i.e., during power-off no key material remains on the device, which reduces the attack surface. Commercially available PUFs include the Loop PUF from Secure-IC[1]; the SRAM PUF from Intrinsic ID[2], which is also used in NXP[3] devices for key storage and by Rambus[4]; the SAMPUF™ [1] by Samsung; and the NeoPUF [2, 3] by ememory[5]. Further vendors provide PUFs in their portfolio without providing details about the exact primitive, e.g., Thales[6] has an analog IP core at hand, and Xilinx[7] provides their Zynq UltraScale+ devices with a built-in PUF to generate a cryptographically strong, device-unique encryption key.

Most importantly, PUFs have to meet quality criteria to assure that the derived keys are different on distinct devices and have full entropy. However, apart from quality criteria and ease of integration into existing solutions, protecting PUFs against Side-Channel Analysis (SCA) is a major concern for practical applications. SCA exploits the fact that ICs may leak information from physical properties such as timing, power consumption or electromagnetic (EM) emanations. In particular, if this side-channel information is related to the processed secret data, SCA is a severe threat that jeopardizes the security of a system. In the context of PUFs, SCA can target the primitive directly, i.e., the circuitry that exploits the manufacturing variations, or the subsequent processing that processes the derived data, e.g., Error-Correcting Codes (ECCs). Understanding potential attacks is a key requirement to develop suitable countermeasures that enable secure key generation from PUFs.

[1] https://www.secure-ic.com/products/issp/security-ip/key-management/puf-ip/, last accessed 22nd December 2022.

[2] https://www.intrinsic-id.com/physical-unclonable-function/, last accessed 22nd December 2022.

[3] https://www.nxp.com/docs/en/application-note/AN12292.pdf, last accessed 22nd December 2022.

[4] https://www.rambus.com/blogs/combining-root-of-trust-and-puf-technology-for-robust-chip-security/, last accessed 22nd December 2022.

[5] https://www.ememory.com.tw/en-US/Products/Product?guid=19081314113656, last accessed 22nd December 2022.

[6] https://www.thalesgroup.com/en/market-specific-solutions/tss/system/physical-unclonable-function-puf, last accessed 22nd December 2022.

[7] https://docs.xilinx.com/r/en-US/xapp1333-external-storage-puf, last accessed 22nd December 2022.

## 1.1 Contributions

This thesis investigates SCA attacks and countermeasures for ring-based PUF primitives, and attacks on BCH codes used for key generation. In particular the following contributions are achieved:

- Practical SCA of the Transient Effect Ring Oscillator (TERO) PUF, demonstrating that attacks are even feasible on primitives with short metastable oscillations.
- SCA and protection of the Loop PUF with sign-based bit derivation. The attack is shown for classical as well as for on-chip side-channel measurements, highlighting the threat and need for countermeasures. The proposed *temporal masking* countermeasure randomizes temporally separated frequency leakage. Deriving the required randomness from the Loop PUF enables a *self-secured* PUF primitive.
- SCA and protection of the Loop PUF with Two-Metric Helper Data (TMHD) bit derivation. It is practically demonstrated that the TMHD scheme can be attacked by analyzing the amplitude of the observed frequency difference, and cannot be protected by temporal masking. Instead, randomizing the bit derivation order in form of *challenge randomization* has to be adopted. Possible lightweight randomization techniques do not provide sufficient protection, and a True Random Number Generator (TRNG)-based countermeasure is very expensive. Therefore, a side-channel hardened modified Loop PUF primitive, the Interleaved Challenge Loop PUF (ICLooPUF), is proposed that avoids exploitable frequency leakage by *challenge interleaving*. A theoretical analysis of the protection mechanism is accompanied by a practical evaluation.
- Extension of SCA of Bose-Chaudhuri-Hocquenghem (BCH) codes in the context of PUF-based error correction. Existing Differential Power Analysis (DPA) attacks are extended, and horizontal SCA attacks are investigated regarding their feasibility and limitations.

## 1.2 Structure

The thesis is organized as follows: In Chapter 2 the generation of secret keys from Physical Unclonable Functions is introduced. In particular, the relevant ring-based PUF primitives for the following chapters as well as bit derivation methods and error correction for post-processing are provided. Additionally, the attacker model and existing physical attacks on PUFs are addressed.

The possibility of attacking the TERO PUF is shown in Chapter 3, which highlights that even ring-based primitives that oscillate for a limited time are susceptible to SCA. Chapter 4 demonstrates that the Loop PUF is prone to classical and remote SCA and requires dedicated protection. Consequently, *temporal masking*, a low complexity countermeasure for the Loop PUF, is proposed, where the temporal order of two challenges used to generate a PUF bit is randomized. As the randomness is drawn from the Loop PUF itself, the result is a *self-secured PUF primitive* with little overhead. Chapter 5 shows that temporal masking does not protect amplitude-based bit derivation. Possible protection mechanisms are analyzed. The analysis leads to a modification on circuit level that interleaves pairs of challenges. The countermeasure denoted as *Interleaved Challenge Loop PUF (ICLooPUF)* hides side-channel leakage, which is theoretically established and practically verified. Chapter 6 relates attacks on ring-based PUF primitives, and compares the countermeasures for the Loop PUF. Further, the transfer of the temporal masking countermeasure to other PUF primitives is explored.

Chapter 7 focuses on possible SCA attacks on BCH codes that are used to correct noisy PUF responses. Improvements of DPA attacks based on modified helper data manipulations and an extended power model are provided that allow for attacking stand-alone BCH codes. As the manipulation of helper data is not always possible, the feasibility of horizontal SCA attacks is investigated. Results from simulated power measurements are provided for both attacks.

Finally, the thesis concludes in Chapter 8.

# 2 Key Generation from Physical Unclonable Functions (PUFs)

This chapter introduces the use of Physical Unclonable Functions (PUFs) for key generation on embedded devices. First, Section 2.1 provides a general overview of the involved entities and their interaction. Subsequently, Section 2.2 summarizes PUF primitives based on rings. A detailed overview of different bit derivation methods is provided in Section 2.3. It is followed by details about error correction that enables a reliable derivation of key material from noisy PUF responses in Section 2.4. Finally, Section 2.5 concludes with the attacker model and existing physical attacks on PUFs.

## 2.1 Overview

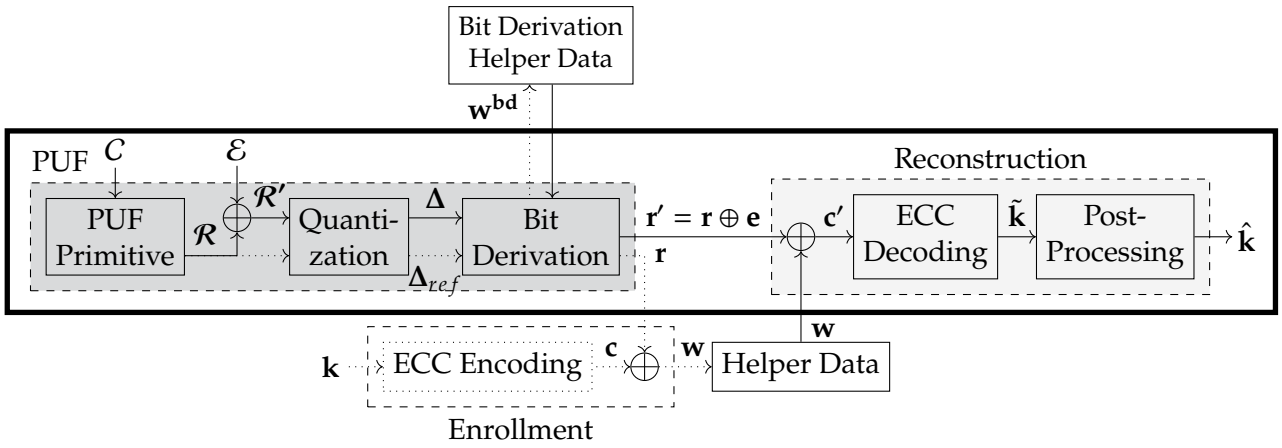



**Figure 2.1** Schematic of PUF-based key generation using the Fuzzy Commitment Scheme (FCS). Dotted lines denote steps that are carried out once during enrollment in a trusted environment, solid lines denote operations required by the reconstruction step in the field.

Fig. 2.1 depicts the general structure of the key generation from PUFs considered in this work that consists of two main building blocks. The Physical Unclonable Function (PUF), highlighted in gray, generates a binary string $\mathbf{r}'$ from analog values $\mathcal{R}$. The PUF response $\mathbf{r}'$ is processed by the Fuzzy Commitment Scheme (FCS) [4] consisting of enrollment and reconstruction, emphasized in lightgray, to derive a reliable key $\hat{\mathbf{k}}$. Further details of Fig. 2.1 and its components are provided in the upcoming sections.

### 2.1.1 PUF: Primitive, Quantization and Bit Derivation

In Fig. 2.1, the PUF is a unit consisting of a PUF primitive, a quantization step and a subsequent bit derivation from the quantized values, i.e., it transforms analog physical variations into binary values. The *PUF primitive* is the source of randomness that captures device-specific manufacturing variations. This means the primitive must be designed such that, while the blueprint of the primitive is identical on different devices, manufacturing variations lead to a device-specific behavior. The primitive is provided with an input, denoted as *challenge* $C_i$, and provides an output termed as *response* $\mathcal{R}_i$. Applying a set of challenges $C \in \{C_0, \ldots, C_{N_C-1}\}$ results in analog responses $\mathcal{R} = \left[\mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_{N_C-1}\right]$. Note that

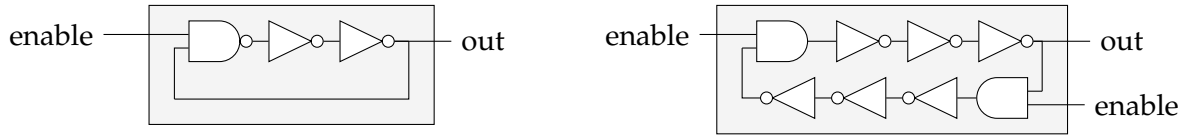
for PUF-based key generation, the challenge set should be fixed and should be provided from within the device, i.e., the challenges are stored on the device or generated on-the-fly; there is no external interface that allows for setting the challenges. The *quantization* transforms the analog responses $\mathcal{R}$ into digitized values $\mathbf{\Delta}_{ref} = \left[\delta_0^{ref}, \delta_1^{ref}, \ldots \delta_{N_C-1}^{ref}\right]$, i.e., each $\mathcal{R}_i$ is mapped to an $m$-bit value $\delta_i^{ref}$. Finally, the *bit derivation* maps the discrete values $\mathbf{\Delta}_{ref}$ to a binary PUF response $\mathbf{r} = [r_0, r_1, \ldots, r_{N_r-1}]$. Note that the number of response bits $N_r$ can differ from the number of challenges $N_C$ depending on the bit derivation scheme, e.g., if more than one response bit is generated from each analog response. Furthermore, depending on the bit derivation scheme that is used, additional side-information, so-called *bit derivation helper data* $\mathbf{w^{bd}}$, about the properties of the quantized values is stored.

The PUF primitive is influenced by environmental conditions such as the temperature and the supply voltage of the device [5, 6, 7]. Furthermore, aging can shift the analog responses with respect to their nominal value $\mathcal{R}$ at the beginning of the PUF's lifetime [8, 9]. Consequently, the analog responses are affected by noise $\mathcal{E}$ and the primitive provides a noisy analog response $\mathcal{R}'_i = \mathcal{R}_i + \mathcal{E}^j$. Note that $\mathcal{E}^j$ is sampled from a distribution each time $j$ an analog response from challenge $i$ is derived, i.e., it is derived from a stochastic process [10]. In Fig. 2.1, the noise term $\mathcal{E}$ encompasses all of the aforementioned noise effects, and additional indices to indicate different samples are neglected for simplicity. After quantization of the noisy analog response $\mathcal{R}'$, the digitized values $\mathbf{\Delta} = \left[\delta_0, \delta_1, \ldots \delta_{N_C-1}\right]$ are combined with the helper data $\mathbf{w^{bd}}$ to derive the binary PUF response. Depending on the bit derivation method and the environmental perturbations, the resulting PUF response $\mathbf{r}' = \mathbf{r} \oplus \mathbf{e}$ can be noisy, i.e., it is perturbed by bit errors $\mathbf{e}$ compared to the response $\mathbf{r}$ that has been generated during enrollment. A reliable key generation is not directly possible if the PUF response $\mathbf{r}'$ contains bit errors as the response differs for each query.

### 2.1.2 Fuzzy Commitment Scheme: Enrollment and Reconstruction

In order to ensure reliable key generation from a noisy PUF response $\mathbf{r}'$, various Helper Data Algorithms (HDAs) exist that differ in the generation of the helper data $\mathbf{w}$, the choice of the secret, and its reconstruction [11]. Similarly to the FCS used in this thesis, the helper data $\mathbf{w}$ of the Code-Offset Fuzzy Extractor (COFE) [12] is computed from the offset of a codeword $\mathbf{c}$ generated from a random number and the PUF response $\mathbf{r}$. However, the PUF response $\mathbf{r}$ is directly used as the secret, and in order to avoid helper data leakage, an additional hash function is required to derive the key $\mathbf{k}$ from the response. On the other hand, in syndrome constructions [12] the helper data is constructed by multiplying the PUF response $\mathbf{r}$ with a parity check matrix $\mathbf{H}$ of an ECC, i.e., the so-called *syndrome* is stored. For reconstruction, the smallest error vector $\mathbf{e}$ is searched that has the same syndrome as the sum of the syndrome of the noisy PUF response $\mathbf{r}'$ and the stored helper data. Compared to COFE and FCS, the helper data size is reduced, but due to possible helper data leakage, the key must be compressed by an additional hash. Parity constructions [13] use the PUF response $\mathbf{r}$ directly as a secret. It is multiplied with the parity part $\mathbf{P}$ of a generator matrix $\mathbf{G}$ from an ECC, and the redundancy is stored as helper data $\mathbf{w}$. Depending on the size of the PUF response and the helper data, information leakage can be an issue, and additional compression of the key by a hash is required. Systematic Low Leakage Coding (SLLC) [14] splits the PUF response $\mathbf{r}$ into two parts, and uses the first part of the PUF response as the secret. The XOR of the parity of the encoded secret part with the second part of the PUF response is stored as helper data, avoiding information leakage from the helper data. Other HDAs as the pointer-based approach Index-Based Syndrome (IBS) coding [15] and its extension Complementary IBS (C-IBS) [16], as well as Differential Sequence Coding (DSC) [17] require knowledge of the statistical properties of the PUF during enrollment.

In this thesis, the *Fuzzy Commitment Scheme (FCS)* [4] is used that embeds a secret key $\mathbf{k}$ into the PUF response $\mathbf{r}$ during an enrollment phase, such that the key can be retrieved from a noisy response $\mathbf{r}'$ in a reconstruction phase. The FCS does not use the PUF response directly as a secret, such that changing the key is possible. Furthermore, post-processing such as compression or hashing is not strictly needed. During the *enrollment phase*, which is carried out once at the beginning of the




**(a)** Stable oscillation

**(b)** Metastable oscillation

**Figure 2.2** Basic configurations of rings used in PUFs.

device's life time in a trusted environment, a secret key $\mathbf{k}$ of $N_k$ bits is generated. The key is encoded by an ECC to a codeword $\mathbf{c}$ that contains redundancy to compensate errors at a later stage. The codeword $\mathbf{c}$ and the PUF response $\mathbf{r}$ are mapped to the helper data $\mathbf{w}$ using an XOR-offset. Note that the ECC is designed such that the $N_k$-bit key results in an $N_r$-bit codeword and consequently the same number of helper data bits $N_w = N_r$. In the field, whenever the key is needed, the *reconstruction phase* allows for generating the same key $\mathbf{k}$ that has been embedded to the PUF response in the enrollment phase. The noisy PUF response $\mathbf{r}'$ and the helper data $\mathbf{w}$ are mapped to a noisy codeword $\mathbf{c}'$ using an XOR-offset. An ECC decoder corrects the noisy codeword to a decoded key $\tilde{\mathbf{k}}$. The probability for a reconstruction error $p_e$ depends on the assumed average bit error probability of the PUF response $p_b$ and the error-correcting capability of the ECC. Usually, the ECC is designed such that a reconstruction error of $p_e < 10^{-6}$ to $10^{-9}$ is reached, i.e., with probability $1 - p_e$ the noise-free key $\tilde{\mathbf{k}} = \mathbf{k}$ is obtained. If the entropy of the PUF response $\mathbf{r}$ and the generated random key $\mathbf{k}$ is sufficiently high, no additional hashing is needed in the FCS, i.e., $\hat{\mathbf{k}} = \tilde{\mathbf{k}}$.

The following sections provide details about ring-based PUF primitives, different bit derivation schemes, and BCH-based ECCs that are used to compensate errors in the PUF response. Finally, possible attack vectors are provided.

## 2.2 Ring-Based PUF Primitives

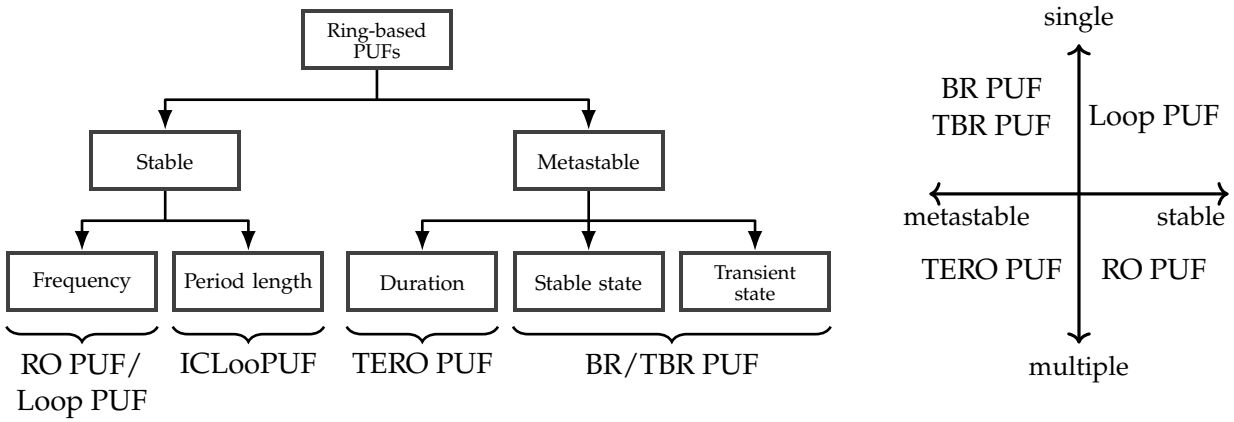
In the two decades since silicon PUFs were introduced in 2002 [18], a variety of PUF primitives has been proposed. The main focus in this thesis is on *ring-based PUF primitives* that consist of a chain of delays or inverting elements with a feedback as depicted in Fig. 2.2. Depending on the number of inverting elements, the ring develops either a stable oscillation that does not stop on its own, or a metastable oscillation that collapses after a certain time.

**Stable and Metastable Oscillations** For an odd number of inverting elements as depicted in Fig. 2.2a, the ring oscillates with a period inversely proportional to the sum of the delays of the individual elements. In order to stop and start the *stable oscillation* in a controllable manner, a NAND gate is used, i.e., setting the enable signal to high initiates the oscillation process, while setting it to low stops the oscillation. If the number of oscillations at the output is measured for a fixed acquisition time $T_{acq}$ by a counter, the counter value is proportional to the oscillation frequency $f$, respectively the period $T$, of the ring:

$$v = \lfloor T_{acq} \cdot f \rfloor = \left\lfloor \frac{T_{acq}}{T} \right\rfloor, \tag{2.1}$$

where the floor operator accounts for the fact that the counter quantizes the frequency. The acquisition time must be sufficiently high, i.e., $T_{acq} \gg T$, to reduce quantization artifacts.

For an even number of inverting elements, as shown in Fig. 2.2b, the output of the AND gate is 0 as long as the enable signal is low. Accordingly, the outputs of the inverters are alternatingly 1 and 0, and the ring takes a stable state, where the second input of the AND gate is 1. As soon as the enable signal is set to high, the two events propagate through the ring. Under ideal conditions the inverters are identical and the events oscillate as long as the enable signal is set to high. However, for inverters with variations the events collapse after some time resulting in a *metastable oscillation*.




**(a)** Evaluation principle **(b)** Architectures

**Figure 2.3** Overview of ring-based PUF primitives.

**Ring-Based PUF Architectures** From the two basic ring configurations different PUF architectures can be derived. Fig. 2.3 provides an overview of ring-based PUFs in terms of their evaluation principles and the configuration of the respective PUF architecture. Two general distinctions can be drawn that are depicted in Fig. 2.3b, namely the stability of the oscillation and the number of oscillators used in the basic architecture.[1]

Following the basic ring configurations from Fig. 2.2, the ring of the primitives can be configured either for stable or metastable oscillation. As depicted in Fig. 2.3a, the *stability* is the main distinction regarding the evaluation principle. On the one hand, constructions with an oscillator that is stable for a defined time, such as the Ring Oscillator (RO) PUF [19] or the Loop PUF [20], usually employ the frequency of oscillations as their secret. Modifications of these primitives, such as the ICLooPUF [21] introduced in Section 5.5, modify the evaluation principle, but still rely on the stability of the oscillations. On the other hand, metastable ring-based primitives derive the entropy from oscillations with an unpredictable duration. Either the duration until the oscillation aborts serves as a secret as for the TERO PUF [22, 23], or the stable or transient state of the ring can be used to derive the entropy, as in the case of the Bistable Ring (BR) PUF [24, 25] and the Twisted Bistable Ring (TBR) PUF [26, 27].

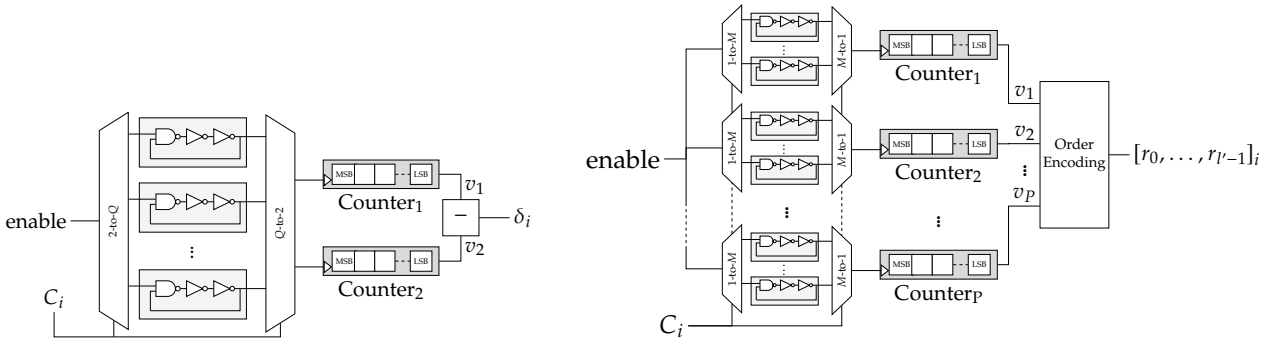
Along the second dimension in Fig. 2.3b, the architectures can be divided into primitives that are based on an array of multiple rings or consist of a single configurable ring. The Loop, BR and TBR PUFs belong to the latter class, while the RO and TERO PUFs can be assigned to the former. Primitives based on ring arrays compare distinct rings with each other, where the number of rings in the array determines the entropy. From a design perspective, spatial gradients are a possible source of bias that have to be considered [28]. In comparison, for primitives based on a single ring, the elements forming the ring are configurable such that more than a single response bit can be derived. The selection of different paths through the ring is achieved by a challenge, where each bit configures a different ring element. Consequently, the length of the ring determines the entropy, which can be extracted. In order to avoid bias effects, the ring design and the evaluation method must assure that neither single elements nor interconnects between the elements dominate the oscillation.

The following sections introduce the PUF architectures distinguished in Fig. 2.3b.

### 2.2.1 RO PUF

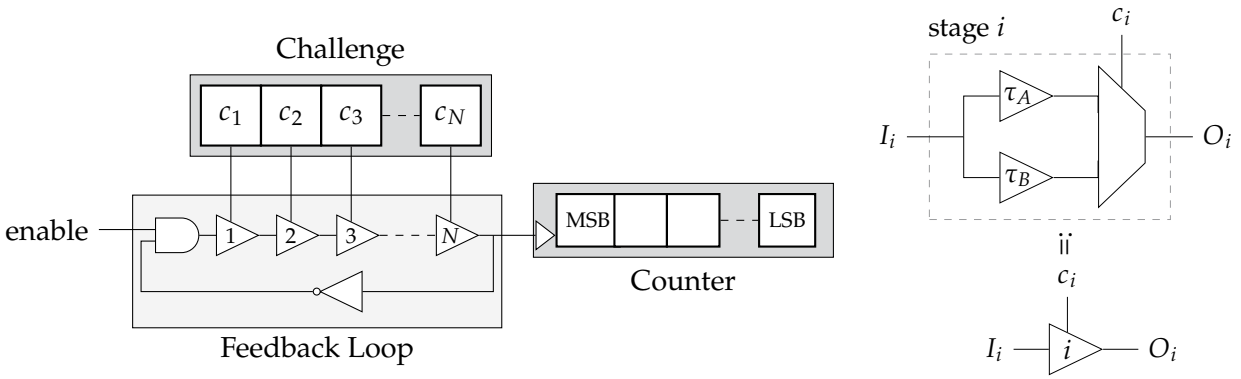
Along with the idea of building silicon PUFs, oscillating circuits were proposed as candidates to characterize ICs, and the comparison of different oscillators to compensate environmental conditions was outlined [18]. Yet, only in 2007 the RO PUF architecture depicted in Fig. 2.4a based on an array

[1] Note that architectures of primitives based on a single oscillator, such as the Loop, BR and TBR PUF, can be extended by using multiple configurable oscillators in parallel.




**(a)** Classical **(b)** PUFKY

**Figure 2.4** RO PUF architectures with (a) a single array of ROs and counter comparison, (b) multiple arrays of ROs and frequency order encoding.




**(a)** Architecture **(b)** Stage design

**Figure 2.5** Schematic of the Loop PUF structure.

of $Q$ oscillators was introduced [19]. It derives PUF bits from the frequency difference of oscillator pairs. Comparing every RO to all other ROs would allow for deriving $\log_2(Q(Q-1))$ bits, which do not have full entropy as the bits are mutually dependent. Overall, the number of ROs in the array, determines the PUF entropy. To achieve an output with full entropy, only non-overlapping pair-wise comparisons can be used. Each oscillator is only used in one comparison, extracting $\log_2(Q/2)$ bits.

Instead of using comparisons of oscillator pairs, the PUF-based cryptographic Key generator design (PUFKY) architecture from Fig. 2.4b is based on $P$ arrays of $M$ ROs [29]. From every array, an oscillator is selected , and the corresponding $P$ frequencies are measured each by a counter. Subsequently, the counter values are ordered and Lehmer-Gray coding is applied to extract an increased number of bits as outlined in Section 2.3.3. Compared to the classical RO PUF, using several arrays that are measured at once, more bits with higher reliability can be extracted [30]. The benefits come at the cost of a hardware overhead due to the increased number of counters.

Note that for both RO PUF designs the selection of the ROs that are measured is considered as the challenge.

### 2.2.2 Loop PUF

The Loop PUF is a ring-based delay PUF with stable oscillation introduced in 2012 [20]. Its main component is a delay chain composed of $N$ identical controllable delay stages. An RO is formed when the output of the delay chain is fed back to the chain's input through an inverting gate. An enable signal allows for starting and stopping the oscillation. Fig. 2.5a illustrates the Loop PUF architecture.

Each of the $N$ delay stages of the Loop PUF contains two delay elements such as inverters or buffers, as depicted in Fig. 2.5b. Due to manufacturing variations, the two elements have different delays $\tau_A$ and $\tau_B$. A challenge bit $c_i$ applied to the $i^{\text{th}}$ stage selects, e.g., via a multiplexer, one of the two elements that is included in the RO path. The challenge $C$ applied to the PUF is the $N$-bit word composed of the $c_i$. The frequency of the RO depends on the sum of the selected delays. Neglecting noise and aging, the frequency is constant for given environmental conditions but unique for each hardware realization of a Loop PUF due to local process variations of the individual delay elements during the device fabrication.

**Operating Mode** The Loop PUF requires an operating mode to derive secret bits from the oscillation frequencies obtained for given challenges. The basic operating mode is presented in Algorithm 1. It consists of two subsequent measurements: The first using the challenge $C$ and the second with the complementary challenge $\neg C$ applied (Lines 1 and 3). In other words, the frequencies of the oscillator are measured with different delay elements in the ring.

**Algorithm 1** Basic Loop PUF Operation

**Input:** Challenge $C$
**Input:** Measurement time in terms of periods $n_{acq}$ of the reference clock
**Output:** Response $\delta_C$ (a signed integer mapped to the secret bit $r_C$)
1: Set current challenge to $C$
2: Count oscillations of the Loop PUF for $n_{acq}$ cycles of the reference clock $\Rightarrow v_C$
3: Set current challenge to $\neg C$
4: Count oscillations of the Loop PUF for $n_{acq}$ cycles of the reference clock $\Rightarrow v_{\neg C}$
5: Compute $\delta_C = v_C - v_{\neg C}$
6: **return** $\delta_C$

The challenge dependent frequency of the RO is the underlying secret, which is measured by counting the number of oscillations of the loop for a fixed predefined measurement time (Lines 2 and 4). For this purpose, the $N$-bit challenge $C$ is applied to the Loop PUF. Then, the enable signal is set to logical 1 while a reference counter counts a predefined number $n_{acq}$ of periods of a reference clock with frequency $f_{ref}$. After the acquisition time $T_{acq} = n_{acq}/f_{ref}$ has passed, the counter value $v_C$ is derived. The counter value $v_{\neg C}$ for the complementary challenge $\neg C$ is derived accordingly. In the original proposal, the sign of the counter difference $\delta_C = v_C - v_{\neg C}$, or equivalently its Most Significant Bit (MSB), is the secret response bit $r_C$ obtained from the Loop PUF [20]. However, other bit derivation schemes, which are introduced in Section 2.3, can be used as well, e.g., in Chapter 5 the TMHD is analyzed.

The differential measurement process compensates for a large amount of influences through environmental conditions and aging effects. Since these effects happen on a larger time scale than the measurement time, successively measured frequencies are affected similarly. Compared to other oscillation-based PUF primitives, such as the RO and TERO PUF, that use multiple rings in their architecture, spatial correlations are avoided by using the same oscillator sequentially.

**Hadamard Challenges for Maximum Entropy** The Loop PUF provides a challenge space of $2^N$ challenges, where each challenge derives a PUF response bit. However, for challenges that have a small Hamming distance, i.e., which are similar, similar delay elements are used and a correlation of the response bits is expected. In the key generation scenario, only a limited number of response bits is required, which have to be independent in order to provide a PUF response with full entropy. In order to get a PUF response with full entropy from an $N$-stage Loop PUF the challenges are chosen as Hadamard codewords from an $N \times N$ *Hadamard Matrix* [31].

Hadamard codewords are pairwise orthogonal and have a Hamming distance of $N/2$ from each other. In other words, for each challenge half of the used delay elements are different compared to

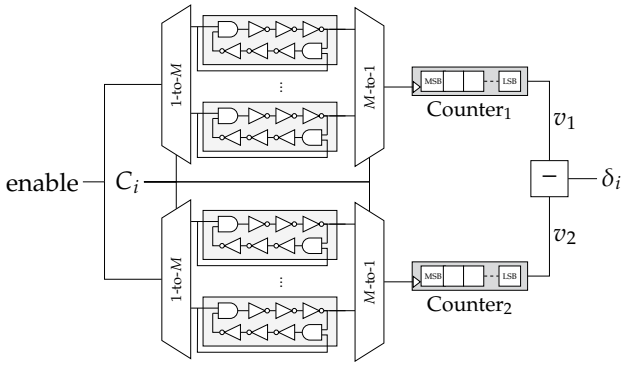



**Figure 2.6** TERO PUF architecture as used in [32, 33, 34].

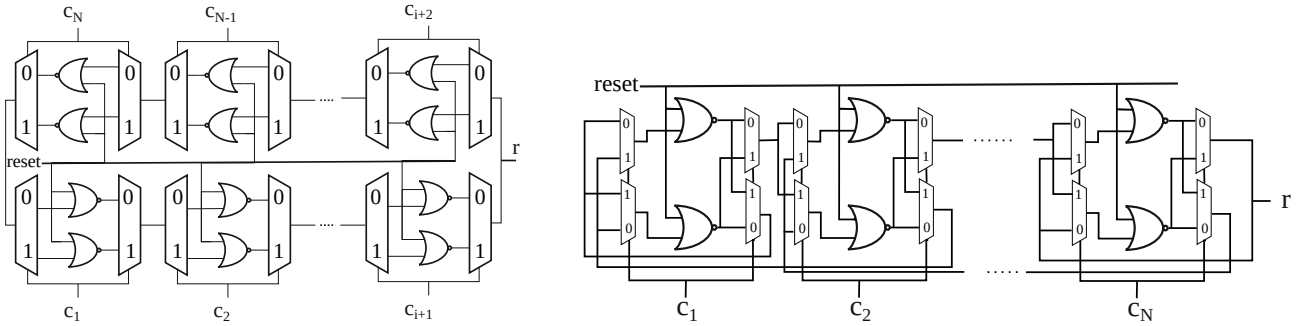
the other challenges, minimizing correlations of the resulting response bits. Furthermore, Hadamard codewords, except for the all-zero and all-one codewords, share a Hamming weight of $N/2$: Half of the delay elements selected by the challenge multiplexer are chosen from the lower and the other half from the upper of the available paths depicted in Fig. 2.5b. In order to avoid systematic bias effects from path delays between the stages, the all-zero and all-one Hadamard codewords are not used and $N - 1$ bits can be derived from a Loop PUF with $N$ stages using Hadamard challenges. Note that Hadamard codewords can be constructed on-chip with low effort, enabling a low-complexity design as only a small memory is required to store the challenges.

### 2.2.3 TERO PUF

The Transient Effect Ring Oscillator (TERO) was introduced in 2010 as an entropy source for TRNGs [35], and the possibility to use it as a PUF primitive was postulated in 2013 [22]. Each TERO cell comprises two identical branches, consisting of an AND gate and an odd number of inverter gates, that form a metastable ring as depicted in Fig. 2.2b. When setting the enable signal from low to high, two events start to propagate. While in theory the TERO oscillates until the enable signal is reset, manufacturing variations of the underlying Complementary Metal-Oxide-Semiconductor (CMOS) structures result in different delays of the two branches and a break down of the oscillation in finite time.

The manufacturing variation dependent number of oscillations until the TERO reaches its stable state has been utilized to construct the TERO PUF in 2014 [23]. In the first proposal, an overlapping comparison of neighboring TERO cells is able to generate 126 bits from 64 TEROs. Later, the architecture shown in Fig. 2.6 became the proposed architecture of the TERO PUF [32, 33, 34]. It consists of two blocks of $M$ TERO cells and two corresponding counters. One TERO cell is selected from each block by a challenge. The two cells are activated and connected to the counters by multiplexers. Thus, only the two TERO cells that are compared oscillate at a time. The selection of pairs of cells is not restricted [32, 33, 34]: Each of the $M$ cells from one block is compared to all $M$ cells from the other block resulting in $M^2$ Challenge-Response Pairs (CRPs). After a fixed acquisition time $T_{acq}$, the activated TERO cells are stopped. $T_{acq}$ allows for a trade-off between reliability, uniqueness, and runtime. It is chosen such that most of the TERO cells are expected to be settled. Therefore, $T_{acq}$ is in the range of several hundred nanoseconds depending on the number of inverters in a branch of the TERO cells, e.g., 600 ns for seven inverters [33, 34].

To derive the PUF response, the counter values after $T_{acq}$ are subtracted. The Least Significant Bits (LSBs) of the difference $\delta_i$ are unstable due to noise and are ignored for the PUF use case, but could be used as a source of entropy for a TRNG. The MSBs, in particular the sign, are variation dependent and relatively stable over time. Hence, the sign bit and specific unique and steady bits (e.g., the fifth to seventh LSBs [33, 34]) serve as PUF response bits. Gray coding can be applied to the difference $\delta_i$ to ease further processing and to potentially increase robustness with respect to noise [32].




**(a)** BR PUF **(b)** TBR PUF

**Figure 2.7** Architectures of the BR and TBR PUF. Adapted from [36].

### 2.2.4 BR PUF and TBR PUF

A Bistable Ring (BR) consist of a ring with an even number of inverters. When released from an unstable state, there are two possible stable states at the output of the inverters, namely '1010 . . . 010' or '0101 . . . 101'. Whether the ring stabilizes within a certain time or keeps oscillating depends on the process variations of the involved elements.

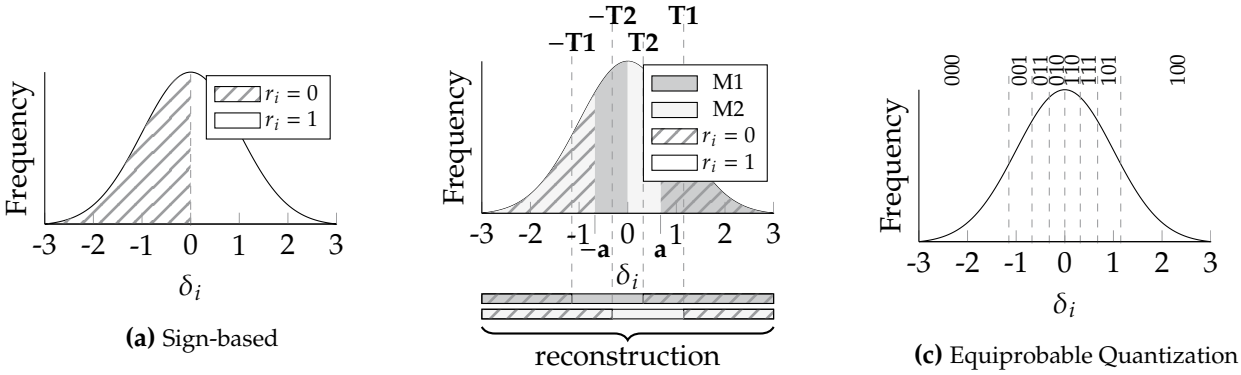
The BR PUF [24, 25] builds inverting elements of two NAND or NOR gates that are selected by a pair of a multiplexer and a demultiplexer. Fig. 2.7a shows a BR PUF with NOR gates, where each of the $N$ inverting elements can be configured by a challenge bit $c_i$; depending on the challenge bit value, the multiplexer and the demultiplexer select the lower or upper NOR gate to be included in the ring. The second input of the gates is connected to a *reset* signal that allows for bringing the ring into an unstable all-zero state, i.e., the outputs of all inverting elements are set to 0 and the ring state is '000 . . . 000'. The ring is started by releasing the reset, such that the gates act as inverters. Due to delay differences of the inverting elements, the propagating signals in the BR collide and the ring state settles to one of the two stable states. The bit of any of the inverting nodes can be used as a PUF bit, i.e., the *stable state* of the settled BR serves as the secret. As the evaluation of the stable state is not possible for BRs with a long settling time, alternatively the *transient state* of the entire ring can be compared to one of the stable states [26]. That is, instead of evaluating the stable state of a single inverting node, the duty cycle of the oscillation is evaluated either at a defined point in time or for several subsequent points in time.

As the CRPs of the BR PUF are linearly related, the Twisted Bistable Ring (TBR) PUF [37] depicted in Fig. 2.7b has been proposed as an improvement. Instead of selecting one of two possible gates in each inverting stage, all inverters are always included and the challenge bits determine the positions of the inverters in the ring. The aforementioned evaluation methods based on the *stable state* or *transient states* can be similarly applied to the TBR PUF.

## 2.3 Bit Derivation From Analog PUFs

In order to derive PUF response bits from digitized analog values, several methods exist that provide different trade-offs regarding reliability, robustness and the number of extracted bits. Following Section 2.2, for most ring-based primitives a counter quantizes the analog values of the oscillator, the BR/TBR PUF being a notable exception. Usually, a comparison of counter values is used to reduce bias effects in the PUF due to offsets caused by local or temporal gradients, e.g., counter values from two ROs for the RO PUF or from two challenges for the Loop PUF are compared. In the following it is assumed that a counter value difference $\delta_i$ is mapped to one or more response bits.

The values are assumed to follow approximately a normal distribution, i.e., the realizations of the random variable $\delta_i$ are drawn from a distribution $\delta_i \sim \mathcal{N}(\mu, \sigma)$ with mean $\mu$ and standard deviation $\sigma$.




**(a)** Sign-based

**(b)** Two-Metric Helper Data

**(c)** Equiprobable Quantization

**Figure 2.8** Bit derivation from analog PUFs. Adapted from ©2022 IEEE [21].

Fig. 2.8 depicts the most common bit derivation methods covered in the following sections for a standard normal distribution with $\mu = 0$ and $\sigma = 1$.

### 2.3.1 Sign-Based Bit Derivation

Fig. 2.8a depicts the sign-based bit derivation, where the sign of $\delta_i$ determines the PUF bit as

$$r_i = \begin{cases} 0, & \delta_i \leq 0 \\ 1, & \text{otherwise} \end{cases}, \tag{2.2}$$

i.e., the MSB of the counter difference determines the PUF bit. The main drawback of the sign-based approach is that it requires a strong error correction: Noise and environmental variation can change the sign of $\delta_i$ if it is close to 0. The changed sign causes faulty bits, as the response from reconstruction deviates from the enrolled value.

**Dark-bit Masking** A simple method to improve reliability is to drop unreliable bits, referred to as *dark-bit masking* [38, 39].[2] In the enrollment phase, multiple measurements are taken to determine the bit error probability $p_{b,i}$ of each bit $r_i$. A bit mask stores for each response bit whether it exceeds a certain error probability, i.e., whether it is considered unreliable and identified as a so-called *dark bit*. For example, in Fig. 2.8a values $\delta_i$ within a certain range $\epsilon_{db}$ around 0, i.e., $|\delta_i| \leq \epsilon_{db}$, are expected to have a higher bit error probability as perturbations from environmental conditions change the sign of $\delta_i$ more easily. The bit mask is stored as bit derivation helper data $\mathbf{w^{bd}}$, and is applied during reconstruction to the PUF response such that dark bits are discarded. On the one hand, dark bit masking leads to an increased reliability of the provided bits as only the most reliable bits are selected. On the other hand, it discards PUF bits and therefore reduces the overall entropy. Note that dark bit masking is not limited to sign-based bit derivation, but can be similarly applied to amplitude-based schemes from Sections 2.3.2 and 2.3.3.

### 2.3.2 Two-Metric Helper Data Scheme

The Two-Metric Helper Data (TMHD) [40] scheme is a method that enhances the reliability of responses that contain reliability information. For ring-based PUFs and sign-based bit derivation, the amplitude provides reliability information: if a value $\delta_i$ is far off the decision bound of 0, it is less prone to errors, while a value that is closer to 0 is more likely affected by errors in terms of a changed sign. Using this information, the TMHD scheme allows for highly reliable responses with

[2] The method of selecting the most robust bits was briefly mentioned for analog outputs of optical PUFs [38], later it was formally described and the term *dark bit* was coined [39].

a decreased use of error correction in the post-processing stage. Furthermore, compared to dark bit masking, no bits are discarded.

The TMHD subdivides the assumed normal distribution of the counter differences into octiles. In the *enrollment phase*, the distribution of all $\delta_i$ on a device is estimated under the assumption that it follows a Gaussian normal distribution with mean $\mu = 0$ and standard deviation $\sigma$, i.e., $\delta_i \sim \mathcal{N}(0, \sigma)$. The octiles are defined by the points $-T1$, $-a$, $-T2$, $T2$, $a$, $T1$ (and $\pm\infty$) as depicted in Fig. 2.8b. 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

$$\Phi(x) - \Phi(y) = \int_y^x \phi(x)dx = \frac{1}{8}. \tag{2.3}$$

The Two-Metric Helper Data (TMHD) method derives its name from the fact that based on the octiles, two metrics $M1$ and $M2$ define the mapping of the frequency difference $\delta_i$ to the PUF bit $r_i$ as:

$$M1 : r_i = \begin{cases} 0, & T2 \leq \delta_i \vee \delta_i < -T1 \\ 1, & -T1 \leq \delta_i < T2 \end{cases} \qquad M2 : r_i = \begin{cases} 0, & T1 \leq \delta_i \vee \delta_i < -T2 \\ 1, & -T2 \leq \delta_i < T1 \end{cases}. \tag{2.4}$$

Note that from the definition of the octiles $\int_{-T1}^{T2} \phi(x)dx = \int_{-T2}^{T1} \phi(x)dx = 1/2$, the values for $r_i$ are equiprobable and no bias is induced by the TMHD. For each bit $i$ the metric is chosen and stored as helper data $w_i^{bd}$, for which the reconstruction from a perturbed value $\delta_i' = \delta_i + \tilde{\delta}$ is more reliable. In other words, the metric is stored as helper data, for which a deviation $\tilde{\delta}$ from the enrollment value is less likely to cause a change of the PUF bit during reconstruction. From Fig. 2.8b 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:

$$w_i^{bd} = \begin{cases} M1, & -a \leq \delta_i \leq 0 \vee a < \delta_i \\ M2, & -a < \delta_i \vee 0 < \delta_i \leq a \end{cases}. \tag{2.5}$$

During the *reconstruction phase*, the distribution and hence the bounds from the enrollment $\pm T1$, $\pm T2$ and $\pm a$ may change due to varying environmental conditions. Therefore, the device measures the distribution of values and estimates a new set $\pm T1'$, $\pm T2'$ and $\pm a'$ for reconstruction. The values $\delta_i'$ are mapped with the metric stored in the helper data $w_i^{bd}$ to the PUF bit $r_i'$:

$$r_i' = \begin{cases} 0, & \left(-T1' > \delta_i' \geq T2' \ \wedge \ w_i^{bd} = M1\right) \vee \left(-T2' > \delta_i' \geq T1' \ \wedge \ w_i^{bd} = M2\right) \\ 1, & \left(-T1' \leq \delta_i' < T2' \ \wedge \ w_i^{bd} = M1\right) \vee \left(-T2' \leq \delta_i' < T1' \ \wedge \ w_i^{bd} = M2\right) \end{cases}. \tag{2.6}$$

For typical noise measurements a Bit Error Rate (BER) of $< 10^{-6}$ is achieved [40], i.e., $r_i' = r_i$ with high probability. Either the correction capability of a subsequent ECC can be reduced or the ECC can be omitted at all, which decreases the overall implementation effort if the TMHD is used.

### 2.3.3 Other Amplitude-Based Bit Derivation Methods

The TMHD is the amplitude-based bit derivation scheme mainly considered in this thesis. However, there are further bit derivation methods that use more than only the sign of digitized PUF values.

#### Equiprobable Quantization

In the context of PUFs, Equiprobable Quantization (EPQ) has been introduced as zero leakage quantization [41] and for tamper-evident PUFs [42]. Similarly to the TMHD scheme, the distribution of analog values is divided into $N_{eqp}$ intervals of equal probability, i.e., for each interval defined by the bounds $x$ and $y$, the probability is

$$\Phi(x) - \Phi(y) = \int_y^x \phi(x)dx = \frac{1}{N_{eqp}}. \tag{2.7}$$

In Fig. 2.8c the Equiprobable Quantization (EPQ) method is depicted for $N_{eqp} = 8$, where each interval $\{\mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_{N_{eqp}-1}\}$ is assigned a symbol from a higher order alphabet $\mathcal{S} = \{s_0, s_1, \ldots, s_{N_{eqp}-1}\}$. While other encodings are possible, using a Gray code, i.e., binary values of neighboring intervals differ by one bit only, reduces the bit errors for small changes in $\delta_i$ [41]. For example, if the interval 010 is enrolled and noise leads to a shift to a neighboring interval during reconstruction, either 011 or 110 is obtained; in both cases only one of three bits changes compared to the enrollment.

During enrollment, the response bits are derived from the interval $j$, where $\delta_i$ falls into, as

$$\left[r_i^0, r_i^1, \ldots r_i^{\log_2\lceil N_{eqp}\rceil}\right] = s_j,\ j \mid \delta_i \in \mathcal{I}_j, \tag{2.8}$$

i.e., as the symbol $s_j$ corresponding to the interval $\mathcal{I}_j$. To increase robustness, the number of intervals (respectively their minimum width) is adjusted to the expected noise level of the PUF during enrollment, and bit derivation helper data $\mathbf{w^{bd}}$ is stored that describes the relative position of the enrolled value within the respective interval. During reconstruction, the knowledge of the relative position from enrollment allows for adjusting the decision boundary. Note that while the interval that a value belongs to is the secret, the helper data of the relative position within the interval does not leak secret information [41].

Compared to the TMHD scheme, the EPQ typically derives more bits, where a certain amount of bits is consumed for redundancy of the error correction. Furthermore, the amount of bit derivation helper data is increased if the granularity of the relative position is more than binary, i.e., more than the upper/lower half of the interval is provided.

#### Order encoding

Instead of deriving bits from single values $\delta_i$ based on intervals of a distribution, *order encoding* takes a set of $P$ values $\{\delta_i, \delta_{i+1}, \ldots, \delta_{i+P-1}\}$ and derives PUF bits from their relative order. In other words, the amplitudes of the values are transformed into a sorted list, which is encoded to a binary string.

In the PUFKY architecture shown in Fig. 2.4b, Lehmer-Gray order encoding has been suggested to derive a large number of stable bits from an array of RO PUFs [29], but could be applied to other ring-based PUFs as well. In the case of the RO PUF, for each position an individual offset from the counter is subtracted in order to remove bias effects for specific oscillator positions. The offsets are determined per device and are stored as bit derivation helper data $\mathbf{w^{bd}}$.

The $P$ different RO frequencies, respectively the corresponding values $\{\delta_i, \delta_{i+1}, \ldots, \delta_{i+P-1}\}$,[3] are sorted with respect to their value. Each of the possible $P!$ orders is assigned a unique bit sequence used as response. The sequence is encoded under the constraint of a low sorting overhead and such that minimal changes in the ordering lead to a small variation in the derived bit sequence, which is fulfilled by Lehmer encoding of the ordering and Gray encoding of the Lehmer coefficients. The number of bits that can be derived from the Lehmer-Gray order encoding is $l' = \sum_{i=2}^{P}\lceil\log_2 i\rceil$, which are usually further compressed to $l \leq l'$ bits to remove bias and bit dependencies [29].

#### Multiple Counter Bits

The commonly used sign-based method from Section 2.3.1, where the MSB of an $m$-bit value is used, can be extended to several MSBs. The method is closely related to Equidistant Quantization (EPQ) [43] that divides the distribution of analog values into intervals of the same width. When deriving a PUF bit from the $j^{\text{th}}$ MSB of the counter difference $\delta_i$

$$r_i(\delta_i[j]) := r_i^j = \begin{cases} 0, & \delta_i \bmod 2^{j+1} < 2^j \\ 1, & \text{otherwise} \end{cases}, \tag{2.9}$$

[3] For the sake of readability and consistency with the other bit derivation schemes, the symbol $\delta_i$, which actually denotes a counter difference, instead of the symbol for the counter value $v_i$ is used. Note that in [29] counter values are sorted, but counter differences could also be used, e.g., for the Loop PUF.

**Table 2.1** BCH codes used for PUF-based key generation.

| | Code construction $(n, k, t)$ | $p_e$ | $p_b$ | key size |
|---|---|---|---|---|
| Bösch et al.[a] (2008) [44] | $25 \times$ [Rep(3, 1, 1)+BCH(63, 7, ,15)] | $8.13 \cdot 10^{-7}$ | 15% | 128[b] |
| | $2 \times$ [Rep(5, 1, 2)+BCH(226, 86, 21)] | $2.28 \cdot 10^{-7}$ | 15% | 128[b] |
| Maes et al. (2012) [29] | Rep(7, 1, 3)+BCH(318, 174, 17) | $\leq 10^{-9}$ | 13% | 128 |
| Van Herrewege et al. (2012) [47] | $7 \times$ BCH(255, 21, 55) | $10^{-6.97}$ | 10% | 128 |
| Van Herrewege/Verbauwhede (2012) [48] | BCH(413, 296, 13) | ?[c] | 3-10% | ?[c] |
| Merli et al. (2013) [49] | $2 \times$ [Rep(7, 1, 3)+BCH(127, 64, 10)] | $10^{-6}$ | 15%[d] | 128 |
| Jarvis/Gaj (2017) [50] | BCH(511, 238, 37) | $4.7 \cdot 10^{-7}$ | 3% | 128 |
| | $44 \times$ BCH(127, 22, 23) | $2.03 \cdot 10^{-11}$ | 3% | 128 |
| SAMPUF™ [1, 46] | $2 \times$ BCH(255, 131, 18) | $2.11 \cdot 10^{-11}$ | 8.62% | 256 |

[a] Only code constructions with BCH codes that result in $p_e \leq 10^{-6}$ are listed as [44] targets $p_e \leq 10^{-6}$.
[b] 171 source bits are decoded and hashing results in 128 bits of key.
[c] Not provided by the authors, and cannot be derived from the provided quantities.
[d] The expected BER is not explicitly provided, but is derived from the other parameters.

the number of bits is increased compared to other methods as more than one bit is derived at once.

Note that for oscillator-based PUFs, the MSBs of the counter difference are related to device-specific variations, while the LSBs are related to the jitter of the oscillator, i.e., can be used to derive randomness. For higher MSBs device-specific and jitter-related terms mix increasingly, i.e., the reliability of the derived bits decreases. Therefore if using several MSBs, it has to be evaluated for each PUF primitive individually how many bits can be reliably derived.

## 2.4 BCH-Based Error Correction for PUFs

The FCS described in Section 2.1 relies on Error-Correcting Codes (ECCs) that encode the secret in the enrollment phase and decode the codeword $\mathbf{c}'$ during the reconstruction phase to compensate bit errors. The choice of the ECC depends on the average bit error probability $p_b$ of the PUF response and the required reconstruction error probability $p_e$, i.e., the probability that the derived key differs from the enrolled key. Different code constructions have been proposed in the context of PUF-based key generation. Instead of simple codes, the concatenation of codes provides the possibility of achieving higher error correction capabilities while maintaining an efficient implementation [17, 44].

In particular, BCH codes are widely used for PUF-based key generation as they can be efficiently implemented in hardware and provide a high error correction capability compared to other codes. BCH codes are linear cyclic block codes with parameters $(n, k, t)$ [45], i.e., the code operates on blocks of data of $k$ bits that are encoded to $n$-bit codewords $\mathbf{c}$. If the number of errors in the received codeword $\mathbf{c}' = \mathbf{c} \oplus \mathbf{e}$ is $|\mathbf{e}| \leq t$, it is guaranteed that the errors can be corrected.

Depending on the Bit Error Rate (BER) $p_b$ of the PUF response, the targeted decoding error probability $p_e$ and the required key size, the code constructions and parameters vary. Table 2.1 provides an overview of BCH-based code constructions that have been proposed for key generation with PUFs. Concatenated codes are a common design choice, where BCH codes are usually used as the outer code and a repetition code provides the inner code. They result in a lower code rate, but allow for using smaller BCH codes that require less resources. However, commercially available PUF constructions such as the SAMPUF™ [1, 46] use stand-alone BCH codes as well.

## 2.5 Physical Attacks on PUFs and Countermeasures

Similar to physical attacks on cryptographic algorithms, where the algorithm can be mathematically sound, but unprotected implementations are prone to attacks, the secret of a PUF that shows excellent quality metrics can be revealed by physical attacks. Consequently, physical attacks are a threat for the PUF's security that has to be studied and mitigated by appropriate countermeasures.

This section systematizes attacker models and their particularities regarding the PUF scenario in Section 2.5.1. Finally, Sections 2.5.2 to 2.5.4 provide an overview of SCA-related attacks on the PUF primitive, the post-processing of its response and further attacks.

### 2.5.1 Attacker Model

The attacker model defines the capabilities of an attacker in terms of access and physical interaction with the device, and the PUF. The PUF-based key generation scenario disposes some peculiarities that are discussed regarding their implications for attacks and countermeasures in the following.

**Attack Vectors** The enrollment phase of the FCS is carried out in a trusted environment. Therefore, attacks are not applicable during the enrollment process. That means, neither the ECC encoding nor the generation of any kind of helper data are a possible attack vector. Consequently, no dedicated countermeasures are required for these steps. However, the attacker is able to perform physical attacks such as side-channel measurements during the reconstruction phase of the PUF. This includes attacks on the ECC decoding step of the FCS as well as the steps required to derive the PUF response: the PUF primitive, and the bit derivation method.

The security of the PUF should not rely on the concealment of the implementation details. For ring-based PUF primitives the number of oscillators, their configuration and the duration $T_{acq}$ for which the ring is activated are considered public. Similarly, the set of challenges $C$ from which the key is generated is considered public. Without further countermeasures, the order of the challenges is not secret either, and can be matched to the order of observations.

**Helper Data Access** In addition to public implementation details, in particular the assumption of public helper data is generally accepted when designing PUF-based key storage: the helper data should not leak about the secret, and storing the helper data in a protected NVM would render the PUF approach useless as the secret could directly be stored [11].

Helper data with read-write access imposes the least constraints on an adversary, i.e., an attacker can read the value of the helper data from enrollment and also change its value in the reconstruction phase. From the possibility of helper data manipulations, various attack vectors emerge that can be hampered by restricting to read-only access. In this case, the helper data is known to the attacker, but modifications are out of scope. Finally, access to the helper data could be further hampered, e.g., if it is stored by fuses88 that are difficult to read out. Limited helper data access represents the worst case scenario for an attacker, but is only relevant on a theoretical level as the security of the system must not depend on the concealment of the helper data.

**Physical Access and Attacker Capabilities** As PUF-based key generation is targeted at low-cost embedded devices that operate in the field, an attacker potentially has physical access to the device including the PUF. The key generation from the PUF will usually be carried out in a startup phase, but an attacker can reset the device to acquire multiple observations of the PUF reconstruction for the same device. Multiple measurements can decrease the Signal-to-Noise Ratio (SNR) by averaging methods or can be used to combine different measurements, e.g., for DPA attacks or for combining information in other manners. The number of measurements $N_t$ may practically be limited by the time and storage capacities of the adversary, but without countermeasures the number of reconstructions is unlimited.

Attacks can be classified into active and passive attacks and into invasive, semi-invasive and non-invasive attacks [51]. For *passive attacks* the device under attack is operated within its specifications and only observations of physical properties are acquired. On the other hand, for *active attacks* the device is operated outside its specifications, e.g., by modifying environmental conditions such as supply voltage, clock supply or temperature. SCA attacks that measure power or EM emanations fall into the category of passive attacks, while clock and voltage glitching or Fault Injection Attacks (FIA) are active attacks. For PUF-based key generation, only passive attacks are considered in the following.

*Power* measurements and *global EM* measurements above the package are *non-invasive* attacks and are the minimum capabilities an attacker with physical access can achieve. At the same time, this kind of attacks poses the least requirements to an attacker and is therefore the most relevant scenario for low-cost devices. This type of measurements acquires the global power consumption, i.e., a mixture of different sources from the design. *Localized EM* measurements, allow for increasing the SNR by placing micro near-field probes in proximity to the part-of-interest of the circuit. When attacking PUF primitives, localized measurements provide additional benefits if component leakage can be spatially resolved, i.e., it can be separated compared to global measurements, where only the combined leakage is observed. Localized EM attacks are semi-invasive, i.e., they 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.

Finally, in a scenario without physical access, the attacker may share the device under attack, e.g., an Field Programmable Gate Array (FPGA) in a cloud server, with the victim. Hence, neither global nor local power measurements from classical equipment are feasible. However, a remote attacker can use *on-chip* sensors to acquire power measurements.

**Implications of Error Correction for Attacks** The fact that an ECC decoder is used in the reconstruction phase lowers the requirements for the attacks as the $(n, k, t)$-decoding module corrects up to $t$ errors in the PUF response. As the implementation details for the PUF-based key generation are public, the error correction module and its parameters are known to the attacker. Hence, if an attack retrieves the PUF response with an error of at most $t$ bits, the erroneous bits can be corrected using the same decoding as the device, and the correct key is retrieved.

### 2.5.2 Side-Channel Analysis and Countermeasures for PUF Primitives

The most direct attack vector is the PUF primitive itself. Several PUF primitives have been targeted by SCA attacks with a focus on optical and local EM semi-invasive attacks. To mitigate the attacks, several countermeasures have been suggested.

**Other PUF Primitives** While the focus of this thesis is on ring-based PUFs, there have been semi-invasive attacks also on other primitives, which are briefly summarized. For Static Random-Access Memory (SRAM) PUFs a cloning attack measures near infrared photonic emissions of the SRAM cells to characterize the PUF and subsequently clone it using a Focused Ion Beam (FIB) [52]. Furthermore, an attack is proposed that exploits the remanesence decay effect of SRAM cells if an attacker is able to overwrite the SRAM used for the PUF [53, 54]. The Arbiter PUF is characterized by analyzing the photonic emissions of the different delay stages in order to deduce a linear model for the Arbiter PUF that can be solved with little effort [55].

**Ring-Based PUFs** The most prominent target from the class of ring-based PUFs is the RO PUF, where several attacks have been carried out:

(*i*) Using Laser Voltage Probing by exposing the backside of a die to an near-infrared laser beam [56]. The intensity of the reflected beam is altered through absorption or interference effects and allows for the recovery of the RO frequencies.

(*ii*) Using localized EM emissions of the ROs over a decapsulated die [57]. Frequencies from simultaneously activated ROs can be identified and exploited if ROs are used in several comparisons, i.e., are activated more than once. Consequently, a possible countermeasures consists in limiting the use of each RO to a single comparison. Additionally, it is suggested to measure multiple, i.e., more than two, ROs in parallel to increase the number of frequencies an attacker has to distinguish.

(*iii*) Using localized EM measurement over a decapsulated FPGA die, single ROs can be resolved if placed far from each other [58]. However, for ROs placed in proximity to each other, separation of single ROs is deemed unlikely. Yet, multiplexers and counters exhibit leakage about the RO frequencies that can be resolved spatially. To impede SCA of counters and multiplexers, measurement path randomization, i.e., using different counters or multiplexers for each evaluation, and interleaved placement of the components are proposed.

(*iv*) Using geometric leaks in the EM spectrum of an ASIC to resolve adjacently placed counters [59]: The RO PUF under attack follows a low-power design to reduce SCA leaks. However, depending on the measurement position on the decapsulated die, the counter frequencies have different amplitudes and can be distinguished. Interleaved placement of components is therefore not sufficient. Parallel comparison of multiple ROs, as proposed by [57], increases the number of possibilities, but does not protect from brute force attacks. Ultra-low-power counters are proposed as a possible hiding countermeasure.

Furthermore, the possibility to identify and locate ROs by their EM emanations has also been studied in the context of TRNGs [60] and for EM cartography in general [61]. Finally, ROs are susceptible to the so-called *locking phenomenon*, where an RO locks into frequencies injected over the supply voltage [62] or by harmonic injection with electromagnetic fields [63]. The latter is most relevant for FIA that could be carried out.

Parallel to the publication of the attack described in Chapter 3 [64], the possibility to observe TERO oscillations and their duration has been exemplified on a decapped chip for a few bits using a spectrum analyzer [65].

### 2.5.3 Side-Channel Analysis and Countermeasures for Post-Processing

Due to noise and imperfections in the PUF response, further post-processing such as error correction and hashing are required in order to derive a reliable key with full entropy. Consequently, the respective steps can become target of SCA.

For Toeplitz hashing, which allows for efficient implementations by combining Linear Feedback Shift Register (LFSR) with an XOR accumulator, a Simple Power Analysis (SPA) attack can distinguish the output of the XOR accumulator [66]. For software implementations of ECCs for PUF-based key generation, naive implementations of BCH and Reed-Solomon (RS) codes are prone to SPA attacks, and template attacks can be mounted as well [67]. Algorithmic processing of the PUF response by hardware implementations of concatenated repetition and BCH codes can be attacked by DPA attacks if the helper data can be manipulated [49]. Improved helper data manipulation can increase the efficiency of the DPA attack further [68].

A *codeword masking* countermeasure has been proposed for linear block codes, such as BCH codes, to hinder DPA attacks based on helper data manipulation [49]. A $k$-bit random mask $\mathbf{m}$ is encoded into an $n$-bit mask codeword $\mathbf{c_m}$ that is added to the actual codeword $\mathbf{c}$ derived from the PUF before decoding. Consequently, the decoder processes $\mathbf{c_m} \oplus \mathbf{c}$ and the side-channel leakage is independent of the secret. For linear block codes, the sum of two code words is a valid codeword and the decoding provides $\tilde{\mathbf{k}} \oplus \mathbf{m}$, where the mask can be removed again.

### 2.5.4 Further Attacks

Apart from SCA attacks, there are other attack vectors that can jeopardize the security of a PUF.

Considering public helper data, a possible attack vector are *Helper Data Manipulation Attacks (HDMAs)*, where an adversary manipulates the stored helper data and observes the system behavior after manipulation [69]. By observing whether a key derivation fails after tailored helper data manipulations, it is possible to retrieve single bits of the PUF response. Sequentially repeating the attack, allows for recovering the entire PUF response and thereby the secret key. Manipulation attacks have been proposed for different RO PUF constructions [70] as well as for Pattern Matching Key Generators [71]. Furthermore, the possibility of HDMA for the DSC decoding scheme has been discussed [17]. Finally, constructions of the Robust Fuzzy Extractor (RFE) [72], which extends the Fuzzy Extractor (FE) against manipulation attacks, can be subject to HDMA if a necessary distance check of the result is omitted [73]. In this case, the reconstructed PUF response can be set to a value known by the attacker. For particular choices of the error correction code even a key recovery is possible. Possible countermeasure to impede HDMAs include hashing of the helper data to limit the possibility of controlled manipulations.

PUFs have also been proposed for means of authentication, where *machine learning attacks* can be applied. In authentication protocols, where the PUF is presented with a challenge and returns a response, the number of challenges and consequently the number of CRPs is not limited. A possible attack consists of collecting CRPs and training a Machine Learning (ML) model that predicts future responses to challenges not contained in the training set, i.e., a model of the PUF is generated [74, 75]. Modeling attacks require several thousands of CRPs to train the model and access to the response, which can be obtained in a protocol setting or via side-channel analysis [76]. In contrast, for PUF-based key storage the collection of CRPs that allow for ML modeling attacks is out of scope as a limited, fixed set of challenges is hard-coded inside the device and the response bits are the secret key. Note that even for key generation, there is the possibility of ML attacks that exploit helper data depending on the error correction code [77]. However, due to the focus on single-challenge key generation, ML attacks using CRPs are out of scope in this work.

Finally, *fault attacks* on the ECC that processes the PUF response have been investigated [78].

# 3 Side-Channel Analysis of the TERO PUF

This chapter describes an SCA attack on the PUF primitive and the bit derivation from Fig. 2.1. In particular, the evaluation principle of the oscillation duration for metastable oscillations from Fig. 2.3 is targeted in combination with a multi-bit derivation from the quantized values. The targeted primitive is the Transient Effect Ring Oscillator (TERO) PUF, an FPGA PUF primitive favored independently by several authors [6, 23, 34]. Breaking the TERO PUF by means of SCA entails specific challenges, e.g., extracting a multi-bit response per TERO cell and measuring the otherwise hard to observe TERO oscillations. Section 3.1 identifies methods to measure the TERO oscillations and introduces a Short-Time Fourier Transform (STFT)-based method for time domain measurements to evaluate the oscillations of the TERO primitive. The experimental setup is introduced in Section 3.2. In Section 3.3, a semi-automatic attack significantly reduces the entropy of the TERO PUF: up to 25% of the response bits are recovered without any error, and the overall error probability of all estimated bits is less than 18%. The estimate of the failure probability for each bit facilitates an optimal smart guessing strategy. Furthermore, assuming a PUF scenario, where up to 20% errors are corrected, the error probability is sufficiently small to consider the examined TERO PUF design with overlapping comparisons broken by the attack. Finally, the number of oscillations can be predicted so accurate that the derivation of multiple bits from a single comparison is prone to SCA attacks.

> The results in this chapter are based on *Tebelmann/Pehl/Immler: "Side-Channel Analysis of the TERO PUF", in* Constructive Side-Channel Analysis and Secure Design*, Springer International Publishing, 2019, pp. 43-60* [64], the first publication to successfully perform an EM-based side-channel attack on the TERO PUF primitive *without* depackaging the chip. In parallel, the possibility to attack the TERO PUF has been exemplified on a decapped chip for a few bits using a spectrum analyzer [65].

**Remarks on the Original TERO PUF Architecture** In the original proposals of the TERO PUF [23, 33], the separation of the two cell blocks is deemed necessary to avoid dependencies of the responses. This may lead to bias in the responses though, as spatial gradients in silicon can cause cells of one part of the die to settle faster (or slower) than on other parts. Comparing adjacent cells would largely counteract such spatial effects. In addition, spatially separating the TERO blocks makes them prone to attacks by localized EM measurements. While resolving adjacent cells may not be feasible by localized EM measurements, identifying spatially separated cells is certainly within scope based on attacks on similar structures [79, 80].

Due to its architectural limits, the TERO PUF in the described form is only suited for PUF-based key generation and *not* for challenge-response authentication like proposed in [34]. From a side-channel perspective, the difference between the scenarios is the control over the challenge of the PUF. For authentication the attacker might have the ability to decide which TERO cells are enabled at a time while for key generation the sequence of applied challenges is fixed or can only be influenced with significant effort. An attacker could possibly control and choose the challenges sent to the PUF in the authentication scenario e.g., over a challenge-response protocol. For key generation, the challenges, i.e., the selected TERO cells, remain constant. Note that challenge-response authentication with a TERO PUF would also open the door for machine learning attacks.

Another important design decision is whether to compare TERO outputs among all TERO cells or not. Comparing a certain cell to multiple other cells enables a similar attack as proposed for RO PUFs [57]. Also, inherent correlations between PUF bits occur and lower the entropy of the remaining

PUF response bits, i.e., the response does not have full entropy as from the comparison of $M$ cells at most $\log_2 M!$ bits of entropy can be extracted [29]. However, restricting the number of derived bits changes the evaluation of the efficiency of the TERO PUF significantly; therefore in the following, the originally proposed design is taken.

## 3.1 Exploration of the TERO PUF

In the following, suitable attack vectors for the TERO PUF are identified in Section 3.1.1. The TERO oscillations are discovered in Section 3.1.2. Subsequently, Section 3.1.3 provides a method for estimating the oscillation duration, and Section 3.1.4 briefly introduces the Short-Time Fourier Transform.

### 3.1.1 TERO Models and Attack Vectors

In order to derive attack vectors and subsequently an attack strategy, it is crucial to understand the working principle of the Transient Effect Ring Oscillator (TERO) and the resulting physical side-channel emanations. Several approaches to physically model the TERO oscillations exist, that aim at providing a stochastic model for the TERO TRNG [81, 82, 83, 84]. However, the results from the physical models are also useful in the context of the TERO PUF. While there is some controversy about the assumption of the different physical models [84], one key observation is that equally built TEROs on a device oscillate with constant and similar frequency [82, 83]. Therefore, TERO cells and their location are identifiable by an attacker based on their characteristic frequency. Another result is that the variation of the duty cycle changes monotonously over time until it reaches 0% or 100% and the oscillation collapses [82, 83]. Consequently, in the spectrum an attacker can observe a TERO cell at a constant and approximately known frequency as long as it is oscillating. The number of oscillations per TERO cell that are used to derive the secret can be estimated from the frequency as soon as an attacker observes the beginning and the end of the oscillation. Depending on the TERO implementation, two attack vectors are considered in the following.

**Multiple usage of TERO cells** A comparison of each cell from one TERO block to all cells of the other TERO block and taking the sign of the counter differences was suggested to increase the number of secret bits [32]. However, an attacker that observes the approximate duration of each of the simultaneously oscillating TERO cells from the two blocks knows which cell is reused. Consequently, the assignment of the observed duration to blocks and, given the public challenge, even to cells is possible. Then, similarly to [57], the attacker knows the sign bit derived from the subtractor (i.e., the secret) from the observed oscillation times and the knowledge which time belongs to which cell.

**Derivation of multiple bits per TERO cell pair** Given that an attacker can estimate the oscillation duration per cell, it is evident that the approximation of the counter difference is possible. Thus, when deriving multiple bits from a TERO cell [23, 33], they are revealed as soon as the attacker observes the oscillations with sufficient precision. An attacker does not have to resolve the counter differences exactly, since only relatively stable bits of the difference can be used. For the difference $\delta = v_1 - v_2$ of two counter values $v_1$ and $v_2$, the PUF bit $r(\delta[j])$ is derived according to Eq. (2.9). The value of the bit derived from the $j^{\text{th}}$ MSB is revealed when counter values are distinguished with a precision of $2^j$ by the attacker. For example, when using Bit 4 and Bit 5, as proposed in [33], an accuracy of at least $2^4 = 16$ is required to learn both bits. This corresponds to a resolution of approximately 85 ns in the time domain for a TERO frequency of 187.5 MHz as found in Section 3.1.2.

Measuring a TERO cell together with exactly one other cell, thereby avoiding the reuse in multiple pairs, improves the situation. It prevents an attacker from resolving the oscillation time to a single cell. Nevertheless, it is still possible to observe the oscillation duration's absolute value. Hence, only the sign bit of the difference remains unknown.

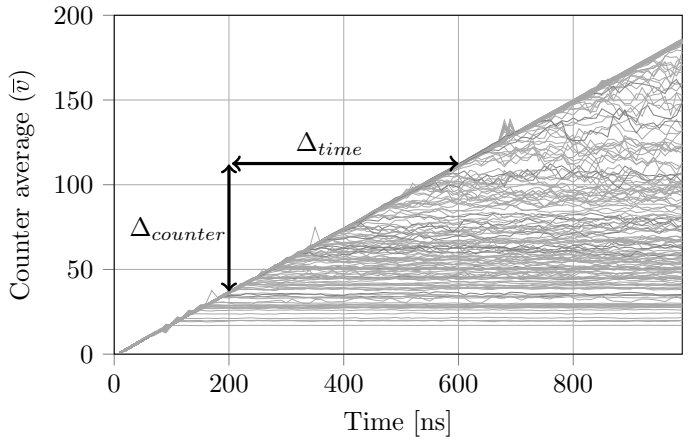



**Figure 3.1** Evaluation of TERO counter values to estimate the oscillation frequency and settling time. Counter values of $2 \times 96 = 192$ TERO cells are averaged over $N = 101$ measurements.

### 3.1.2 Discovering TERO Oscillations

An accurate estimate of the number of oscillations is the basis for the presented attacks. The major obstacle is the short oscillation time until a TERO cell settles. Fig. 3.1 depicts a practical evaluation of the settling time from counter values of different TERO cells. The acquisition time in the experiment is varied from 1 to 99 clock cycles at a clock frequency of $f_{clk} = 100\,\text{MHz}$, corresponding to 10 ns to 990 ns. The respective counter values of each TERO cell are averaged over 101 measurements to compensate for noise at room temperature. The break point of each curve in Fig. 3.1 indicates the end of the oscillation of the corresponding TERO cell, i.e., its settling time.

The results show that most of the TERO cells settle within less than 600 ns. This emphasizes the need for a good time resolution in order to observe the duration of the oscillation. It also motivates the acquisition time of 600 ns used in Section 3.3, which is selected according to [34]. Note that a longer acquisition and oscillation time is beneficial for the attack.

In addition to the settling time of the TERO cells, the slope in Fig. 3.1 confirms that TERO cells indeed oscillate with similar and constant frequency until the oscillation breaks down [82, 83]. The expected frequency

$$f_{TERO} = \frac{\Delta_{counter}}{\Delta_{time}} \approx 187.5\,\text{MHz} \tag{3.1}$$

for the considered design is derived from the slope of the counter values, where $\Delta_{counter}$ is the difference of the counter for a given difference $\Delta_{time}$ of the time. Note that the experiment is only carried out to validate the assumption of a constant oscillation frequency. For an attack, it is not necessary to obtain frequency counter values, as the frequency can be estimated by observing the spectrum.

### 3.1.3 Estimating TERO Oscillation Durations

The experiment in Section 3.1.2 shows a stable oscillation of the TERO cells until the oscillation breaks down. The actual frequency of a certain TERO cell is approximated by $f_{TERO}$ and typically lies in a small interval $f_{TERO} \pm f_\Delta$. Note that the nominal oscillation frequency $f_{TERO}$ is device specific, and $f_\Delta$ is the relative deviation for a certain TERO cell on the same device. In other words, all TERO cells oscillate with $f_{TERO}$ on average, but small deviations $f_\Delta$ are possible for individual cells. From the frequency $f_{TERO}$ and the time $T_{osc}$, for which a TERO cell oscillates before it settles, the number of oscillations $N_{osc}$ is estimated as

$$N_{osc} \approx f_{TERO} \cdot T_{osc}. \tag{3.2}$$

As the counter counts the number of oscillations, its value $v$ is proportional to $N_{osc}$. Due to smaller duty cycles towards the end of the oscillation, counter value corruptions can occur as counter Flip-

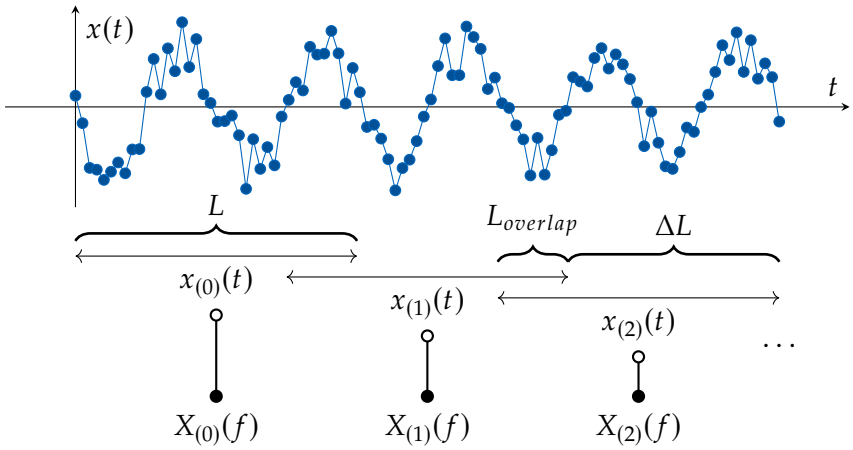



**Figure 3.2** Short-Time Fourier Transform. The windowing function $w(t)$ is omitted for simplicity.

FLOPS (FFs) are only guaranteed to function for a duty cycle of 50% and have differing critical timing characteristics. Spikes exceeding the slope in Fig. 3.1 can be attributed to counter value corruptions [84]. All TERO cells are activated for an acquisition time $T_{acq}$ of $n_{acq}$ clock cycles of the system clock with frequency $f_{clk} = \frac{1}{T_{clk}}$. Therefore, the counter value, respectively the number of oscillations, is upper bounded by

$$v \sim N_{osc} \leq (f_{TERO} + f_\Delta) \cdot T_{clk} \cdot n_{acq} = (f_{TERO} + f_\Delta) \cdot T_{acq}, \tag{3.3}$$

where equality is reached iff the TERO oscillates until the acquisition time $T_{acq}$ ends. Note that for many practical cases the relative frequency deviation $f_\Delta$ of individual cells is negligible in Eq. (3.3). For a deviation $f_\Delta$ from the nominal oscillation frequency $f_{TERO}$, the difference between the actual counter value $v$ and the estimated counter value $\hat{v}$ for $T_{osc} \leq T_{acq}$ is at most

$$|\hat{v} - v| \approx |(f_{TERO} \pm f_\Delta) \cdot T_{osc} - f_{TERO} \cdot T_{osc}| \leq |f_\Delta| \cdot T_{acq}. \tag{3.4}$$

For example, for an accurately known $T_{osc} = T_{acq} = 600\,\text{ns}$ and $|f_\Delta| \leq 1.67\,\text{MHz}$ the difference between actual and estimated counter value is 1 when $f_\Delta$ is neglected. This is below the expected variations in the measurement due to noise.

Summing up, as long as the counter frequencies lie within a narrow range around the nominal frequency $f_{TERO}$, the approximation in Eq. (3.2) holds and the counter values are indeed estimated by the oscillation time $T_{osc}$.

### 3.1.4 Short-Time Fourier Transform

To estimate $T_{osc}$, the visibility of the TERO frequency $f_{TERO}$ in the spectrum is analyzed. The challenge regarding the measurement is the short acquisition time of $T_{acq} = 600\,\text{ns}$ and oscillation times as short as $T_{osc} = 100\,\text{ns}$. In order to resolve time and frequency simultaneously, a STFT-based approach is taken.

Each time domain signal $x(t)$ during the activation of a TERO cell is processed via the STFT, depicted in Fig. 3.2, into the frequency domain. Instead of transforming the entire signal, segments $x_{(l)}(t)$ of length $L$ are taken from $x(t)$, where $(l)$ denotes the index of a segment. Each segment is multiplied by a Hanning window (raised cosine) function $w(t) = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi t}{L-1}\right)\right]$ to reduce spectral leakage effects, where the samples in each segment are defined as $t \in \{0, \ldots, L-1\}$. Windowed segments are transformed individually into the frequency domain:

$$X_{(l)}(f) = \text{FFT}\left(x_{(l)}(t) \cdot w(t)\right) = \text{FFT}\left(\hat{x}_{(l)}(t)\right). \tag{3.5}$$

The segments overlap for a number of samples $L_{overlap}$. In other words, the segments are shifted by $\Delta L = L - L_{overlap}$ samples along the time axis. From the preliminary evaluation of the TEROs in

Section 3.1.2 the frequencies of the TERO cells are stable. This allows for averaging in the frequency domain for each segment over $N_t$ measurements per cell to enhance the SNR.

In order to eliminate signals such as the system clock and other disturbances, a noise floor can be estimated to facilitate the evaluation. For the noise floor estimate, measurements $n(t)$ are taken while the TERO cells are deactivated and the same processing as in Eq. (3.5) is applied. Averaging over $N_{noise}$ measurements yields the noise frequency spectrum $\bar{N}_{(l)}(f)$ for each segment $(l)$.
From the averaged signal $\bar{X}_{(l)}(f)$ and averaged noise floor $\bar{N}_{(l)}(f)$

$$\bar{X}_{(l)}(f) = \frac{1}{N_t} \sum_{i=1}^{N_t} \mathrm{FFT}\left(\hat{x}_{i,(l)}(t)\right), \quad \bar{N}_{(l)}(f) = \frac{1}{N_{noise}} \sum_{i=1}^{N_{noise}} \mathrm{FFT}\left(\hat{n}_{i,(l)}(t)\right) \tag{3.6}$$

the frequency and segment dependent average SNR of segment $(l)$ is defined as

$$\overline{\mathrm{SNR}}_{(l)}(f) = 10 \log\left(\frac{\bar{X}_{(l)}(f)}{\bar{N}_{(l)}(f)}\right) = 10 \log\left(\bar{X}_{(l)}(f)\right) - 10 \log\left(\bar{N}_{(l)}(f)\right). \tag{3.7}$$

The proposed attack evaluates the SNR around the expected TERO frequency $f_{TERO}$. During the period of activation, $\overline{\mathrm{SNR}}_{(l)}(f_{TERO})$ is expected to take higher values. Estimating the time of the activation period translates into measuring the duration of the high SNR.

Note that estimating the noise floor is not a premise for the attack. Instead of evaluating the relative changes defined by the SNR in Eq. (3.7), absolute values of Eq. (3.5) could be used to carry out the attack.

**Frequency resolution** For a real valued signal $x(t)$, the spectrum is symmetric and can be reduced to $N_{FFT}/2 + 1$ bins ranging from Direct Current (DC) to $f_{max}$. Given the sampling frequency $f_s$, the resolution in the frequency domain, i.e., the frequency between two bins, is

$$\Delta_{FFT} = \frac{f_{max}}{N_{FFT}/2} = \frac{f_s}{N_{FFT}} \tag{3.8}$$

with $f_{max} = f_s/2$ being the maximum frequency that can be reconstructed according to the Shannon-Nyquist theorem. In general, a narrow frequency resolution $\Delta_{FFT}$ is desired, which can be achieved by keeping the sampling rate as low as possible, i.e., $f_s \geq 2 \cdot f_{TERO}$ or by increasing the number of Fast Fourier Transform (FFT) bins $N_{FFT}$.

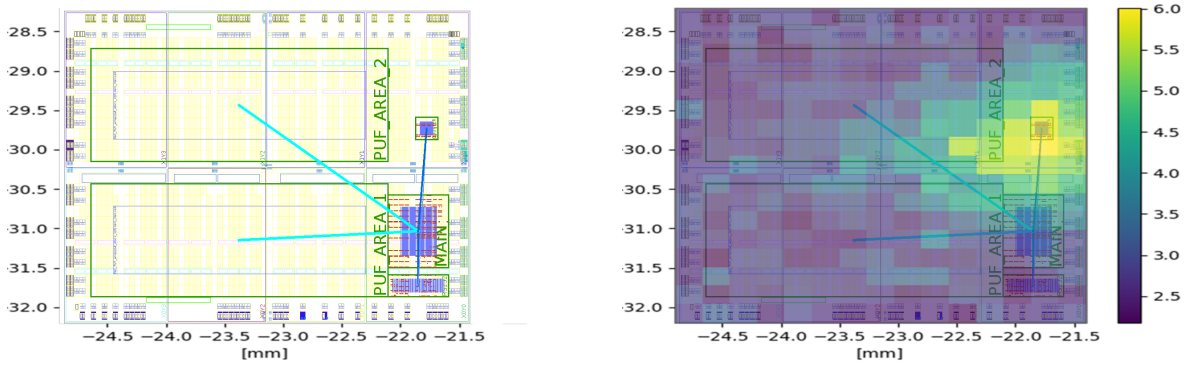
**Temporal resolution** An attacker is mostly interested in the duration of the signal, i.e., the TERO frequency $f_{TERO}$ does not have to be resolved in detail and a trade-off towards the temporal resolution is acceptable. The temporal resolution also depends on $N_{FFT}$ and $f_s$ and behaves contrary to the frequency resolution, where again a narrow value is generally desired:

$$\Delta_T = \frac{N_{FFT}}{f_s} = \frac{1}{\Delta_{FFT}}, \tag{3.9}$$

i.e., to get a good temporal resolution, high sampling rates are required. Without overlapping segments, the resolution would be too coarse to analyze the TERO oscillations. As the segments overlap by $L_{overlap}$ samples, a certain redundancy between segments exists, i.e., since the same samples are transformed, the resulting amplitudes are similar. Yet, as the oscillations stop after some time, all segments that contain samples during the oscillation provide information, and smaller differences than $\Delta_T$ can be resolved as shown in Section 3.3.

## 3.2 Experimental Setup

In the following, the experimental setup is described in terms of the measurement setup, the design under attack, and a pre-evaluation by means of EM cartography.




**(a)** Floorplan (rotated)

**(b)** EM cartography

**Figure 3.3** Design under attack and EM cartography. (a) Floorplan of the TERO PUF on the Spartan-6 FPGA, (b) heatmap with maximum SNR in the frequency range 180-200 MHz during the first 60 ns.

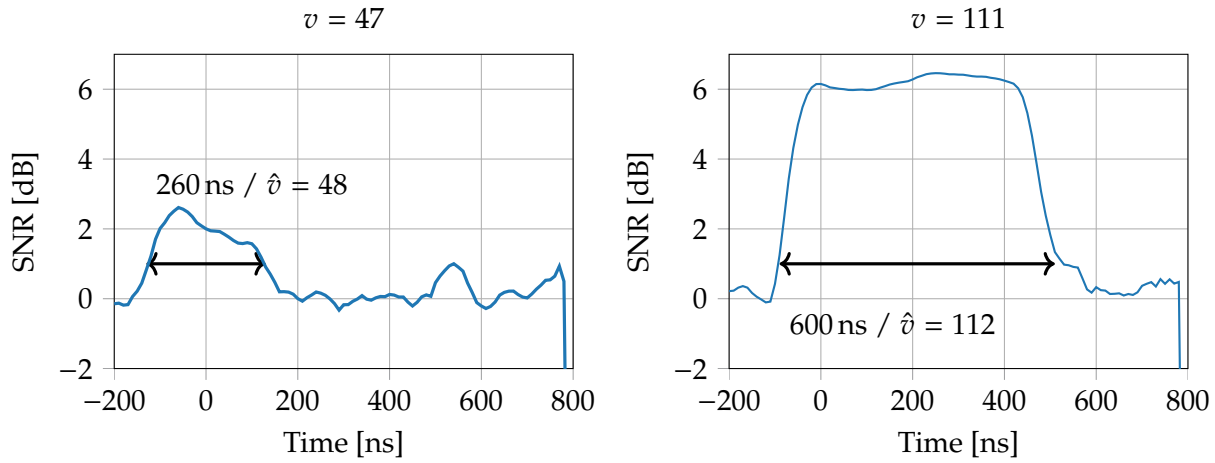
**Measurement Setup** Measurements are recorded with an oscilloscope of 2.5 GHz analog bandwidth and a sample rate of $f_s = 20\,\text{GS/s}$. The near-field probes *RF-B 0.3-3* and *RF-B 3-2* from Langer EMV are used, having $< 1\,\text{mm}$ and $\approx 1\,\text{mm}$ resolution respectively. Both probes capture emanations in vertical direction relative to the FPGA package. Two 30 dB amplifiers amplify the time domain signals. Following the trade-off between frequency and temporal resolution from Section 3.1.4, the number of FFT bins is set to $N_{FFT} = 4096$ resulting in a frequency resolution of $\Delta_{FFT} \approx 4.88\,\text{MHz}$ and a temporal resolution of $\Delta_T = 204.8\,\text{ns}$ per segment ($l$). A segment offset of $\Delta L = 200$ samples, corresponding to a full clock cycle length of the system clock with $T_{clk} = 10\,\text{ns}$, is selected as a trade-off between computational cost and temporal resolution. In all experiments in Section 3.3 the number of noise measurements to estimate the noise floor is set to $N_{noise} = 9600$ and the SNR is evaluated by its maximum in the range from 180 MHz to 190 MHz, corresponding to $f_{TERO} \approx 187.5\,\text{MHz}$ from Section 3.1.2.

**Design Under Attack** The evaluated target is a Xilinx Spartan-6 LX16 FPGA in a 324-pin BGA package mounted on a Nexys3 development board, where the package of the FPGA remains unaltered. The design under attack contains two blocks of $M = 96$ TERO cells each. For the TERO cells a hard macro[1] for the Spartan-6 is used with seven inverters per branch [33]. The number of cells per block is slightly reduced compared to the original TERO PUF proposal in order to include serial communication and a Finite State Machine (FSM) on the same chip.

Fig. 3.3a depicts the floorplan, where the TERO blocks are denoted as `PUF_AREA_1` and `PUF_AREA_2` respectively. The logic for selecting specific cells in each block and assigning their output to the counters is contained in `MAIN`, located in the lower right corner. The counters are placed separately adjacent to the second block of TERO cells. The separation allows to verify whether EM emanations stem from the TERO cells or the counters. The counters are placed side by side to prevent spatial separation of their EM emanations. This thwarts attacks targeting each counter separately.

**Pre-evaluation with EM cartography** Fig. 3.3b depicts a heatmap overlay with the floorplan obtained by using the *RF-B 0.3-3* probe and an xyz-table. The maximum SNR in the frequency range from 180 MHz to 200 MHz is shown. Measurements are taken on a grid of $0.25\,\text{mm} \times 0.25\,\text{mm}$ over the part of the package where the die is located. Each point is measured $N_t = 10$ times while a cell from each block is activated. Similarly, the noise floor is measured from $N_{noise} = 10$ traces. The SNR according to Eq. (3.7) is evaluated during the first 60 ns after a trigger signal. According to Fig. 3.1, in this period all cells oscillate and no settling effects take place. The maximum SNR in Fig. 3.3b coincides with the location of the counters in the design. The area spans almost $1\,\text{mm}^2$, i.e., a fine-grained

[1] `https://perso.univ-st-etienne.fr/bl16388h/salware/tero_puf.htm`, last accessed 29th July 2022.




**(a)** Short oscillation **(b)** Long oscillation

**Figure 3.4** Example SNRs for separately activated cells. Double arrows depict the estimated oscillation duration.

search over the package is not needed and the *RF-R 3-2* probe is positioned manually for all following experiments. Note, that the counter emanations are in line with previous work on EM analysis of ROs [49] showing that observed EM emanations are most likely caused by multiplexers, counters and wires in between. This is no limitation of the proposed attack, since – similar to the attack on ROs [57] – it mainly exploits that TERO cells are used for multiple comparisons.

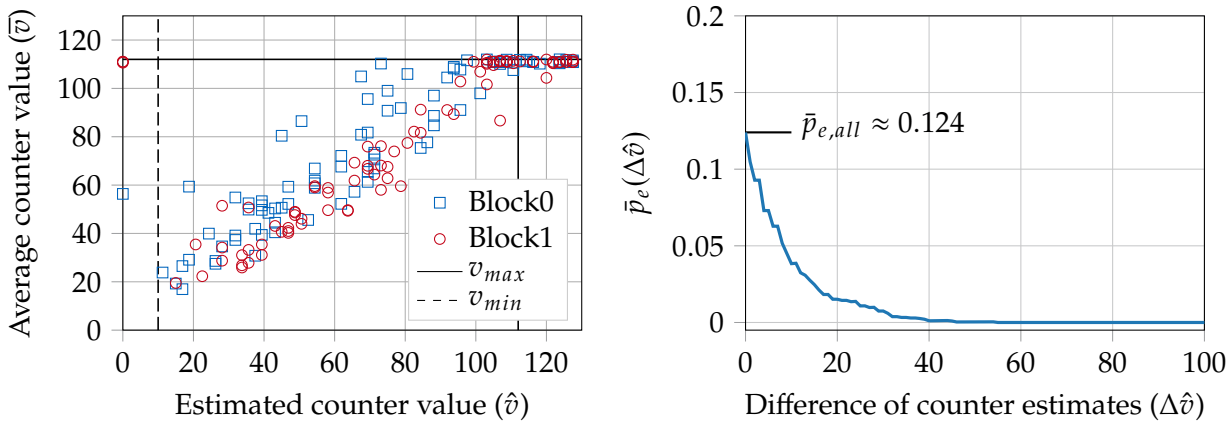
## 3.3 Exploitation of the TERO Side-Channel

This section demonstrates that TERO PUFs are vulnerable to non-invasive SCA. Section 3.3.1 shows the feasibility of detecting TERO oscillations by activating cells separately. Subsequently, Section 3.3.2 practically exploits the reuse of a certain cell in the derivation of multiple response bits, i.e., two cells under comparison are activated at once. The results illustrate that the oscillation duration is well estimated by the STFT approach. A simple countermeasure is not to reuse a certain TERO cell in multiple comparisons. The analysis in Section 3.3.3 nevertheless shows that if multiple bits are extracted from a single comparison still some counter values are leaked which renders the extraction of more than one bit per TERO cell pair insecure.

### 3.3.1 Analysis of Separately Activated Cells

In this experiment only one cell is activated at once to verify the practicality of the STFT approach outlined in Section 3.1.4. Fig. 3.4 depicts the maximum SNR while shifting the segment under transformation in the time domain. The frequency range from 180 MHz to 190 MHz is chosen according to $f_{TERO}$ from Section 3.1.2. The point where the first sample of the segment in the time domain is aligned with the starting point of the oscillation corresponds to 0 ns. Note that also segments starting before 0 ns can include samples at which the TERO cell oscillates. Thus, the increase in SNR starts before this point in time is reached. In addition, cells with an oscillation time shorter than the FFT window can cause a maximum before 0 ns.

The activation of cells causes an increase of the SNR in Figs. 3.4a and 3.4b. At approximately −100 ns, i.e., when the oscillation starts in the middle of the segment under transformation, the SNR reaches 0.75 dB. This value is chosen as a threshold to estimate the oscillation duration $T_{osc}$ as the time from exceeding the threshold to falling back below this value. Assuming an oscillation frequency $f_{TERO} = 187.5$ MHz for all TERO cells, the counter values are computed according to Eq. (3.2) as $\hat{v} = 48$ and $\hat{v} = 112$, respectively. The result fits well to the actual counter values $v = 47$ and $v = 111$, i.e., both short and long oscillations of the TERO are well estimated by the STFT approach.




**(a)** Counter estimation vs. real value

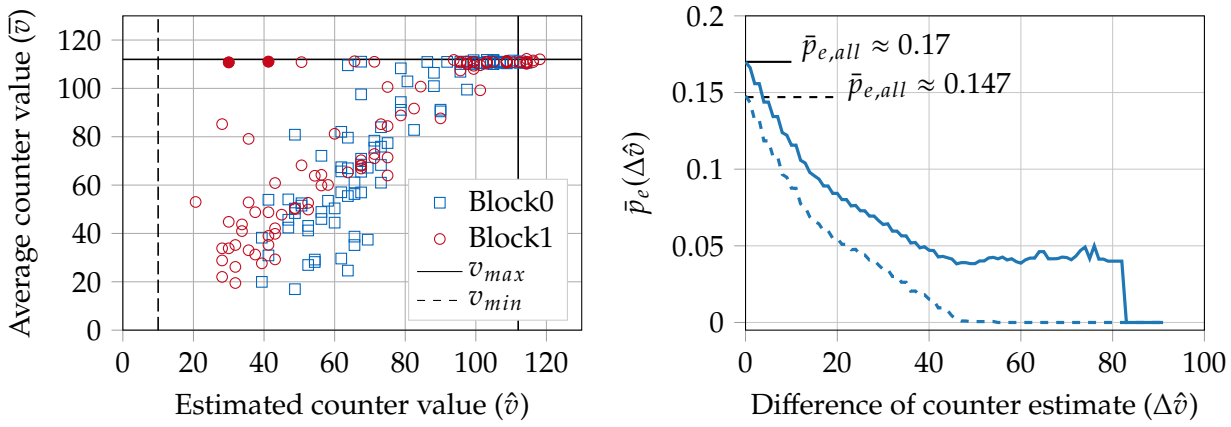
**(b)** Error of estimated TERO PUF bits

**Figure 3.5** Results for the attack on the TERO PUF for separately activated cells. (a) Automatically estimated vs. actual counter values. (b) Probability of guessing a wrong bit using the estimates in (a).

A comparison of estimated and actual counter values for all TERO cells is depicted in Fig. 3.5a. The actual counter values, which slightly vary due to noise, are derived from averaging among $N_t = 100$ measurements. Since the acquisition time, for which the TERO cells are activated, is set to $T_{acq} = 600\,\text{ns}$, the maximum number of oscillations is $v_{max} = f_{TERO} \cdot T_{acq} \approx 112$. Thus, estimated values $\hat{v} > v_{max}$ can be assumed to be $v_{max}$. Indeed, in Fig. 3.5a, almost all estimated values $\hat{v} > v_{max}$ correspond to the maximum possible value, i.e., the minor overestimation of oscillations does not affect the result. Since it is known that all TERO cells have a minimum oscillation duration, certain values below $v_{min}$ are known to be false, e.g., estimates $\hat{v} = 0$ are physically not possible.

**Entropy reduction of the TERO PUF** Comparing each cell from one block to all cells of the respective other block and taking the sign bit as a secret results in $M \times M = 96 \times 96 = 9216$ response bits. From the $2 \times 96 = 192$ estimations of the oscillation duration, four results are deemed unreliable as $\hat{v} < v_{min} = 10$, i.e., for the corresponding $4 \times 96 = 384$ bits no estimation can be given. For the remaining 8808 bits, the probability of guessing the PUF bit erroneously depends on the difference $\Delta\hat{v} := \left|\hat{v}_i - \hat{v}_j\right|$ of the estimated counter value $\hat{v}_i$ from block 0 and $\hat{v}_j$ from block 1 as depicted in Fig. 3.5b. The graph shows the probability of an error $\bar{p}_e$ if only difference estimates $\Delta\hat{v}$ greater than the value on the x-axis are considered. Clearly, the error probability decreases with an increase of the differences. This is in line with Fig. 3.5a: The deviation in the average counter value (ordinate) of the scatter plots is an indicator of the estimation accuracy. An inaccurate estimation has more impact if the estimated counter values are close to each other compared to when the estimated counter values are further apart. According to Fig. 3.5b estimated counter differences with $\Delta\hat{v} \geq 55$ have an error probability of $\bar{p}_e = 0$, i.e., the 2368 bits corresponding to these estimates are revealed without any errors. Estimated counter differences with $\Delta\hat{v} \geq 19$ still have an error probability of only $\bar{p}_e \approx 1.5\%$, which applies to 5471 bits. The whole set of 8808 bits has an error probability of $\bar{p}_{e,all} \approx 12.4\%$.

Summing up, for automatic estimation of single TERO cell oscillations and using only known error free bits, the entropy of the TERO PUF is reduced by a quarter from 9192 to 6848 bits. In addition, an attacker can take advantage from error probabilities for estimations. They define a confidence for each bit that allows to develop a smart guessing strategy, i.e., the remaining guessing effort is below an exhaustive search. Also, an attacker can try to adjust the counter values for counters which contribute to unreliable differences, e.g., by visual inspection of the SNR, which provides more precise results than automatic estimation.




**(a)** Counter estimation vs. real value

**(b)** Error of estimated TERO PUF bits

**Figure 3.6** Results for the attack on the TERO PUF for two simultaneously activated cells. (a) Automatically estimated vs. actual counter values. (b) Probability of guessing a wrong bit using the estimates in (a); dashed line: result for manually discarding estimates marked in solid red in (a).

### 3.3.2 Analysis of Simultaneously Activated Cells

Following the preliminary experiment of attacking single cells, this section analyzes the scenario that two cells are activated at once: each TERO cell in block 0 is compared to each cell in block 1, where the two cells under comparison are activated in parallel. In this setting, each cell $i$ is measured $M = 96$ times, always in combination with a different cell $j \in \{1, \dots M\}$. Assuming that an attacker can figure out which cell is activated at a certain point in time, e.g., by knowledge of the design, the attacker averages over the SNR of all $M$ measurements for cell $i$. For all measurements of cells $i$, the same cells $j$ are compared. Thus, contributions from the other cells result in an approximately constant, distinguishable offset $\overline{\mathrm{SNR}}_{(l)}(f)$, that can be considered as noise:

$$\mathrm{SNR}_{(l)}(f) = \frac{1}{M}\sum_{j=1}^{M}\mathrm{SNR}^{i,j}_{(l)}(f) = \frac{1}{M}\sum_{j=1}^{M}\mathrm{SNR}^{i}_{(l)}(f) + \mathrm{SNR}^{j}_{(l)}(f) \tag{3.10}$$

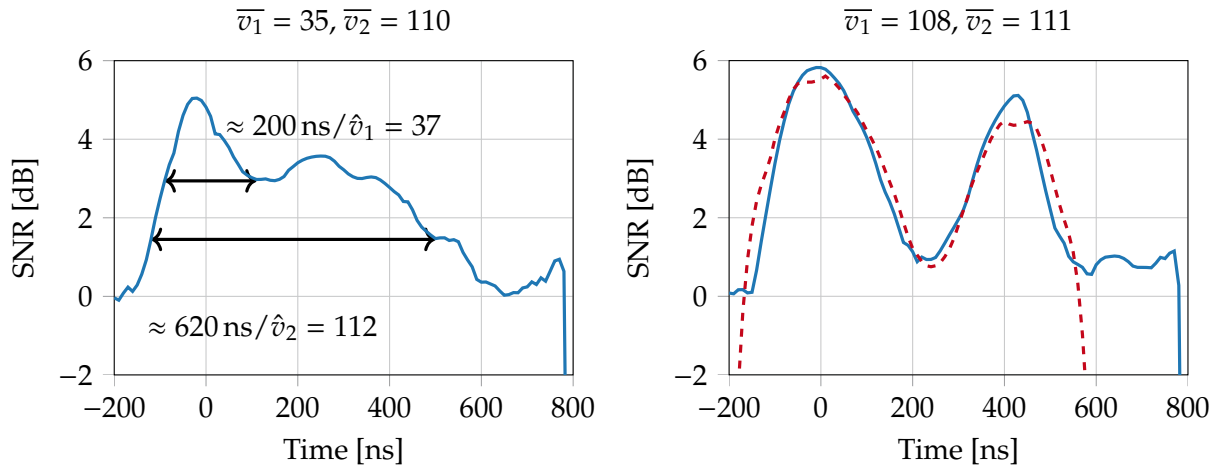
$$= \mathrm{SNR}^{i}_{(l)}(f) + \frac{1}{M}\sum_{j=1}^{M}\mathrm{SNR}^{j}_{(l)}(f) \approx \mathrm{SNR}^{i}_{(l)}(f) + \overline{\mathrm{SNR}}_{(l)}(f). \tag{3.11}$$

.

Effectively, the scenario of activating two cells at once is transformed back to the case of separately activated cells. Due to the activation of two cells and an additionally observed noise floor of approximately 1 dB, the threshold in the automatic counter value estimation is increased from 0.75 dB to 2.5 dB. Fig. 3.6 depicts the results for cells of both blocks. For every comparison, a single measurement is taken and the noise floor is subtracted. The $N_t = M$ measurements of comparisons containing the same cell are averaged, i.e., the number of traces per cell is in the same range as in the previous experiment.

Fig. 3.6a compares automatically estimated counter values against averaged known counter values for this scenario. As expected, the results are slightly degraded compared to Fig. 3.5a, since not all effects caused by simultaneously activated cells cancel out. Due to few but substantial deviations of automatically estimated counter values from the actual ones, the resulting error probability for guessing bits in Fig. 3.6b increases to $\bar{p}_{e,all} \approx 17\%$ compared to $\bar{p}_{e,all} \approx 12.4\%$ in Fig. 3.5b. While still more than 7600 out of 9216 bits are guessed correctly, this reduces the confidence of the guess for almost all bits.

To significantly improve the result, the underlying SNR is evaluated to eliminate cases showing a distorted SNR over time when compared to Fig. 3.4. The most obviously degraded cases in the results




**(a)** Visual estimation **(b)** Model-based estimation

**Figure 3.7** SNR over time for two simultaneously activated cells ($N_t = 100$). (a) visually estimated oscillation duration. (b) additional model given as red dashed line.

correspond to the two solid red dots in Fig. 3.6a. Eliminating these yields the dashed line for the error probability in Fig. 3.6b. Similar to Fig. 3.5b, bits that are guessed with an estimated counter difference of $\Delta\hat{v} \geq 62$ are regarded error free. This applies to 831 bits while smart guessing can be used to get remaining bits as suggested above.

In a typical PUF setting, an error correction is applied to the PUF response to compensate variations due to e.g., environmental conditions. Hence, an BER in the estimated response of up to the correction capability is tolerable for a successful attack. Please note, despite an empirical BER in the range of 5 to 10% within academic settings [5, 6], it is common practice to consider a substantially larger amount of errors in a commercial setting to ensure a failure free operation throughout the whole lifetime of a product, including industrial temperature ranges from −40 °C to 85 °C, leading to an anticipated BER of 15 to 20%. Therefore, the examined TERO PUF can be considered broken by the STFT-based attack even if not all bits are known.

### 3.3.3 Attack on Multi-Bit Responses

The attack in Section 3.3.2 is prevented by using every TERO cell in only one comparison such that a cell is only compared to one cell from the other block. Averaging over multiple measurements, as if a cell is compared to different other cells, is not possible if only pair-wise comparisons are allowed. Then, an attacker cannot assign a counter value to a certain cell as long as the measurements cannot spatially resolve the cells. Consequently, an attacker cannot reveal the sign bit of counter differences. However, if multiple bits are derived from a comparison of the counters, the difference itself is of interest, since knowing its absolute value reduces the entropy to one bit, as discussed in Section 3.1.1.

While no automatic detection is implemented, visual inspection of the SNRs reveals the difference of counter values in many cases as depicted in Fig. 3.7a. Knowing from the previous investigations that the SNR over time develops a plateau while a TERO cell is oscillating, decreases afterwards, and has a knee when no more oscillations are seen in the time segment under observation, the graph can be interpreted: The first peak corresponds to the duration of the first oscillation, while the second oscillation is present until the plateau decays and the SNR vanishes in the noise floor of approx. 1 dB. In Fig. 3.7a the counter values, and thus the difference, is estimated quite accurately and only the sign bit is still secret when neglecting unreliable LSBs.

In contrast, Fig. 3.7b shows that revealing the counter differences from the SNR is more difficult in other cases. Still, by modeling the behavior of the TERO, the two apparently untypical peaks are explained: (*i*) The two TEROs have similar oscillation durations, but (*ii*) in the model the TERO with

the shorter oscillation duration has a 1.5 MHz higher oscillation frequency. Property (*i*) causes that the end points of the oscillation durations can hardly be distinguished, while (*ii*) results in a cancellation of the oscillations in the spectrum due to the relatively coarse resolution of measurements in the frequency domain. The model prediction is marked by the red dashed line in Fig. 3.7b.

The results from Section 3.3.2 and Section 3.3.3 prove that methods to derive multiple bits per pair of TERO cells are vulnerable to side-channel attacks. Similarly, the attack applies to multi-bit derivation from the classical RO PUF (c.f. Fig. 2.4a) as it shares the same architecture, i.e., if not only the MSB of the counter is used. In case of the RO PUF, the amplitude of the frequency difference could be observed from SCA measurements, reducing the entropy for an attacker to the sign bit.

# 4 Self-Secured PUF: Protecting the Loop PUF by Masking

This chapter analyzes side-channel vulnerabilities of a PUF primitive from Fig. 2.1 with stable oscillations in combination with a sign-based bit derivation. The targeted primitive, the Loop PUF from Section 2.2.2, is an area efficient PUF implementation with a configurable delay path based on a single RO. In Section 4.1, SCA results from power and EM measurements confirm that oscillation frequencies are easily observable and distinguishable, breaking the security of unprotected Loop PUF implementations. Furthermore in Section 4.2, on-chip measurements acquired by a Time-to-Digital Converter (TDC) sensor indicate that remote SCA attacks may be feasible in multi-tenant FPGA scenarios. In order to thwart SCA, a low-cost countermeasure denoted as temporal masking is presented in Section 4.3 that requires only one bit of randomness per PUF response bit. The randomness is extracted from the PUF itself creating a *self-secured PUF*. The concept is highly effective regarding security, low complex, and has low design constraints. Finally, Section 4.4 discusses possible trade-offs of side-channel resistance, reliability, and latency.

The results in this chapter are based on the publications *Tebelmann/Danger/Pehl: "Self-secured PUF: Protecting the Loop PUF by Masking", in* Constructive Side-Channel Analysis and Secure Design*, Springer International Publishing, 2020, pp. 293-314* [85] and *Tebelmann/Wettermann/Pehl: "On-Chip Side-Channel Analysis of the Loop PUF", in* Proceedings of the 2022 Workshop on Attacks and Solutions in Hardware Security*, Association for Computing Machinery, 2022, pp. 55-63* [86].

## 4.1 Classical Side-Channel Analysis of the Loop PUF

This section provides the methodology and results for the SCA of the Loop PUF. First, the implementation of the Loop PUF and its use in the experimental setup is described in Sections 4.1.1 and 4.1.2. Subsequently, methods to detect the Frequencies of Interest (FoIs) at which the Loop PUF oscillates are proposed in Section 4.1.3 and a side-channel attack is conducted in Section 4.1.4.

### 4.1.1 Loop PUF Implementation

The most sophisticated part of a Loop PUF from Section 2.2.2 is the implementation of the delay chain. Ideally, the expected delay of the Loop PUF is independent of the challenge, and a difference in the

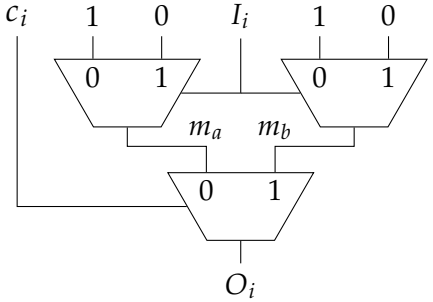



**Figure 4.1** Sketch of the Lookup Table (LUT) utilization of a Loop PUF stage.

delay is only due to process variations affecting the delay elements. In other words, wiring should have no influence and the delay elements in a delay stage according to Fig. 2.5b should be as similar as possible.

To reach this goal, the Loop PUF implementation in this work utilizes the multiplexer structure of the Artix-7 FPGA in accordance to the suggestions for an area-efficient RO PUF design in [87]: Every slice of the Xilinx Artix-7 FPGA contains four 6-input-2-output LUTs. The inputs to a LUT select a path from functionality dependent initialized SRAM cells through a multiplexer tree to the LUT output. Fig. 4.1 sketches the concept for a delay element implemented in a 2-input-1-output LUT. To implement two distinct inverter gates as the basic delay elements (alternatively buffers can be realized) of a delay stage in one LUT, the SRAM at the input of two multiplexers in the same hierarchy level is initialized so that their outputs ($m_a$, $m_b$) correspond to the inverse of a certain input ($I_i$). An additional challenge input ($c_i$) selects if the LUT output is $O_i = m_a$ or $O_i = m_b$. Consequently, the routing between delay stages from $O_i$ to $I_{i+1}$ is independent of the challenges and does not influence the delay differences.

For $c_i$ and $I_i$, inputs of the LUT are selected such that the *expected* delay is independent of the challenge bit. Still, due to the FPGA's internal routing and the implementation of the path from SRAM cells through multiplexers to the output, a certain challenge dependent systematic delay bias might be caused. This corresponds to delay elements in Fig. 2.5b, which are faster or slower on all devices and would result in a reduced entropy of the Loop PUF. If the same amount of fast and slow paths are active for the challenges which are compared, i.e., for the challenge $C$ and the inverted challenge $\neg C$, the effect is mitigated assuming all LUTs are affected by the same systematic effect. Challenges $C/\neg C$, which are selected correspondingly, have the same Hamming weight. For challenges that are Hadamard codewords, this property is inherently fulfilled if the all-zero challenge $C_0 = [0 \dots 0]$ is discarded.

From the described delay elements, a 64-stage Loop PUF is implemented in only 17 slices in eight Configurable Logic Blocks (CLBs). The Loop PUF is realized within a closed domain with fixed placement and routing such that it does not interfere with other parts of the design. The other parts of the design are placed in a separate area but without additional constraints regarding placement and routing. Using Hadamard codewords and discarding $C_0$, the design generates 63 bits.

A single Loop PUF on the device corresponds to the best case for an attacker. Using multiple Loop PUFs in parallel, e.g., to increase the amount of PUF bits in a key-storage scenario, the attacker faces the additional obstacle of spatially resolving different counters, which has been shown to be feasible using localized EM measurements [59]. The additional barrier of localized measurements does, however, not change the overall results and is deemed out of the scope of this work. To further support the analysis, the design supports supplying challenges externally and reading back the measured counter values allowing for validation of side-channel leakage. Responses are computed on a PC receiving the counter values from the device, since the analysis does not consider the potential leakage in the comparison step. In a practical scenario the attacker is not required to have access to any of the internal counter values or being able to apply challenges.

### 4.1.2 Experimental Setup

The experimental setup for the SCA evaluation of the Loop PUF consists of a ChipWhisperer 305 Artix FPGA Target (CW305), that features an Artix-7 (XC7A100TFTG256) running at $f_{clk}$ =100 MHz. A PicoScope 6402D USB oscilloscope performs the acquisition at a sampling frequency of $f_s$ =1.25 GHz. The input bandwidth of the scope is 250 MHz, which is sufficient regarding the oscillation frequencies of the Loop PUF and their harmonics that are in the range from 15 MHz to 65 MHz as shown in Section 4.1.3. Measurements are performed in parallel for the power and EM side-channels as depicted in Fig. 4.2; power measurements are acquired using the SMA jack `X4` of the CW305, which outputs the voltage drop of the FPGA's internal supply voltage $\mathtt{VCC_{int}}$ over a 100 mΩ shunt amplified by a 20 dB low-noise amplifier. EM measurements are taken using a Langer EMV RF-R 50-1 near field probe with a diameter of approximately 10 mm. A 30 dB Langer EMV PA303 pre-amplifier is

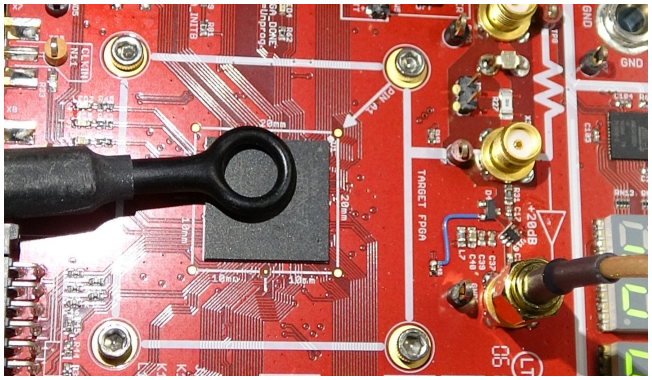



**Figure 4.2** Measurement setup for the SCA evaluation of the Loop PUF. The *RF-R 50-1* EM probe position and a SMA cable connected to the power jack **X4** of the CW305 board are depicted.

used to enhance the signal amplitudes in order to benefit from the oscilloscope's dynamic range. The EM probe is placed on the front-side about 1 mm above the package to capture field lines that are orthogonal to the package surface. A coarse positioning procedure is applied to find the location of interest above the package: For each quadrant of the package measurements are taken, and the procedure in Section 4.1.3 is used to determine whether the relevant frequencies are present. The position providing the highest peak at the frequency of interest is chosen for all further evaluations.

### 4.1.3 Frequency of Interest Detection

In order to attack the Loop PUF, an attacker has to determine the frequencies of the oscillation termed as Frequency of Interest (FoI) in the following. In Fig. 4.3 the spectral representation of different detection methods are depicted. All figures are based on a single measurement per challenge, where the Loop PUF is activated for $T_{acq} \approx 5.24\,\text{ms}$. The first 5.2 ms are transformed into the frequency domain using a FFT of $N_{FFT} = 2,684,359$ frequency bins and a Hanning window to minimize aliasing effects. The resulting spectra exhibit various spikes which makes automatic evaluation difficult. Thus, low-pass filtering is applied along the frequencies to smooth the spectrum. Using the filtering technique, single frequency noise form perturbations and artifacts are reduced, while Loop PUF frequencies, that have a small fluctuation, remain.

Figs. 4.3a and 4.3b show the spectra $X(f)$ of two challenges $C$ and $\neg C$ for power and EM measurements respectively. The oscillation frequency is approximated from the counter value $v_C$ as

$$f_C \approx f_{ref} \cdot \frac{v_C}{n_{acq}} = \frac{v_C}{T_{acq}}, \tag{4.1}$$

with $f_{ref} = f_{clk}$. Note that due to the discrete counter values, $f_C$ is subject to quantization noise. The Loop PUF frequency $f_0 \approx 15.77\,\text{MHz}$, verified by Eq. (4.1), is indicated as well as the multiples $f_1, \ldots, f_3$. In the power side-channel, the frequencies show notable peaks, while in the EM side-channel, peaks are partly covered by other signals. Furthermore, in both side-channels, frequency peaks unrelated to the Loop PUF show up. While some frequency components can be attributed to expected sources such as the system clock $f_{clk} = 100\,\text{MHz}$, other frequencies are a priori indistinguishable from the Loop PUF frequency. Therefore, two methods for reliable Frequency of Interest (FoI) detection are proposed.

**FoI Based on Signal-to-Noise Ratio** The first method subtracts an estimated noise floor $N(f)$ from the spectra $X(f)$, generating a Signal-to-Noise Ratio $\text{SNR}(f) = {}^{X(f)}/_{N(f)}$ similar to the approach in Section 3.1.4. Results are depicted in Figs. 4.3c and 4.3d. The noise floor is estimated from measurements with an inactive Loop PUF, eliminating certain irrelevant frequencies, such as the clock frequency. In Figs. 4.3c and 4.3d the noise floor estimate $N(f)$ is based on averaging over the

frequency spectra of 128 measurements, where the Loop PUF was not active. Compared to the spectra $X(f)$ in Figs. 4.3a and 4.3b, the frequencies $f_1, f_2, f_3$ show up more clearly in the SNR($f$) and other frequency components are canceled out. The basic frequency $f_0$ is covered by other signals in the EM side-channel. The peaks at 68.6 MHz and its multiple at 137.2 MHz are unrelated to the Loop PUF, yet the candidate frequencies for an attacker are reduced.

**FoI Based on Standard Deviation** An attacker may not be able to estimate the noise floor reliably by idle measurements, e.g., if other operations, which are not active in the idle measurements, run in parallel to the Loop PUF. Thus, a second FoI detection method is proposed based on the standard deviation over frequency spectra of all challenges. The basic idea is that frequency components present in all measurements, such as the clock frequency, show a low standard deviation, while frequencies that vary for different measurements produce a higher standard deviation. In Figs. 4.3e and 4.3f, the standard deviation of the frequency spectrum among the different challenges is depicted for power and EM measurements. Indeed, the FoI detection in the power side-channel in Fig. 4.3e reveals the Loop PUF frequency $f_0$ as well as multiples $f_1, f_2, f_3$. In the EM side-channel, Figs. 4.3e and 4.3f, a frequency ramp is visible between 15 MHz and 24 MHz, that partly covers $f_0$. Thus, the fundamental Loop PUF frequency of $f_0$ can still be sensed with a priory knowledge, but is hardly identifiable for an attacker. Only $f_1, f_2, f_3$ are clearly visible. Similar to the SNR-based method, additional frequencies are detected around 68.6 MHz and 72 MHz that are unrelated to the Loop PUF. Overall, more unrelated peaks occur compared to the SNR-based method, but FoIs can be more clearly distinguished compared to the raw spectra in Figs. 4.3a and 4.3b.

Concluding, two methods to detect the FoIs are proposed that allow for determining the frequencies related to the Loop PUF. If possible, the SNR-based method is preferable, otherwise calculating the standard deviation across challenges provides sufficient information.

### 4.1.4 Side-Channel Analysis of the Loop PUF

The frequencies in the range of the FoIs determined in Section 4.1.3 are evaluated regarding the possibility of extracting information about the Loop PUF. The following evaluations focus on a spectral range from 31.4 MHz to 31.7 MHz, because a frequency around 31.54 MHz is identified as a FoI in the EM side-channel. The same frequency range is used for power side-channel to ease comparison.

As noted in Algorithm 1 from Section 2.2.2, the counter value $v_C$ that results from the challenge $C$ is compared to the counter value $v_{\neg C}$ that results from the complementary challenge $\neg C$. The challenges are applied sequentially, thus an attacker can observe the frequencies $\hat{f}_C$ and $\hat{f}_{\neg C}$ separately. If the order in which $C$ and $\neg C$ are applied is known, as is the case for the design presented in Section 2.2.2, the attacker can guess the PUF bit $r_C$ by comparing the frequency spectra of the challenges.

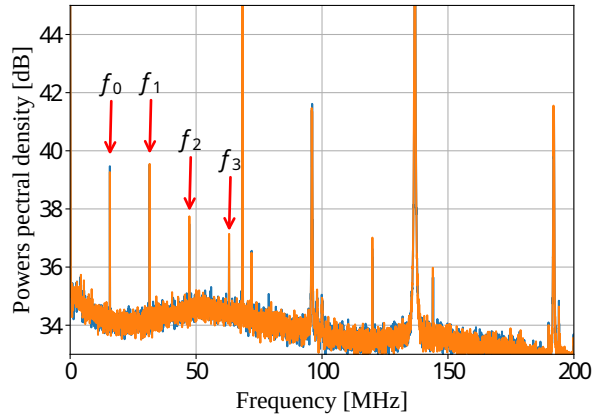
In Fig. 4.4 the typically observed spectra for challenge $C$ and its complement $\neg C$ are depicted. The peaks $\hat{f}_C$ and $\hat{f}_{\neg C}$ are clearly different and can be distinguished. The sign of the comparison $\Delta\hat{f} = \hat{f}_C - \hat{f}_{\neg C}$ is used as the guess for the PUF response bit, i.e.,

$$\hat{r}_C = \begin{cases} 1 & \text{if } \operatorname{sign}(\Delta\hat{f}_C) \geq 0 \\ 0 & \text{if } \operatorname{sign}(\Delta\hat{f}_C) < 0. \end{cases} \tag{4.2}$$

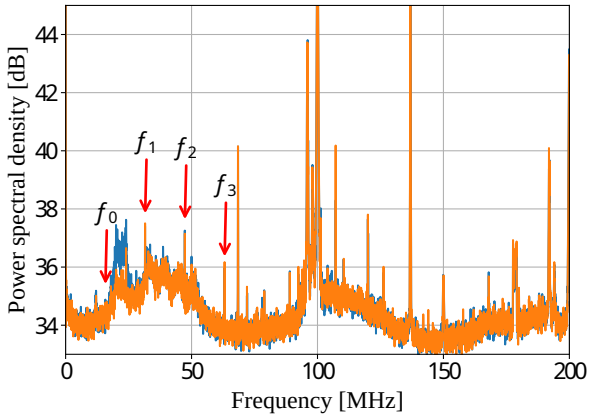
In order to determine the success of an attack on all Loop PUF bits, the actual counter difference $\Delta v_C = v_C - v_{\neg C}$ is compared to its estimate

$$\Delta\hat{v}_C = \left\lfloor \hat{f}_C \cdot T_{acq} \right\rfloor - \left\lfloor \hat{f}_{\neg C} \cdot T_{acq} \right\rfloor \tag{4.3}$$

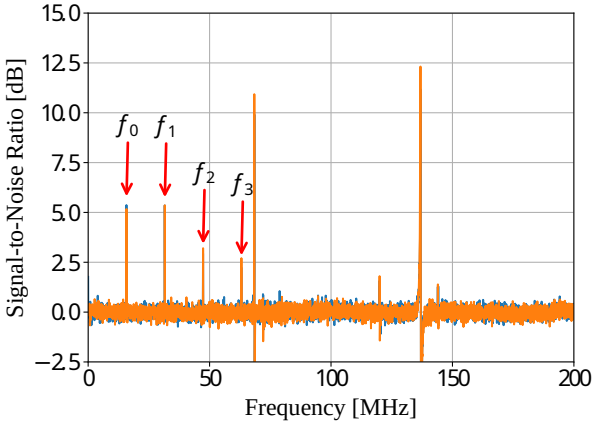
determined from the side-channel observations. The floor operator reflects the assumption that the counter value is incremented after every Loop PUF oscillation.



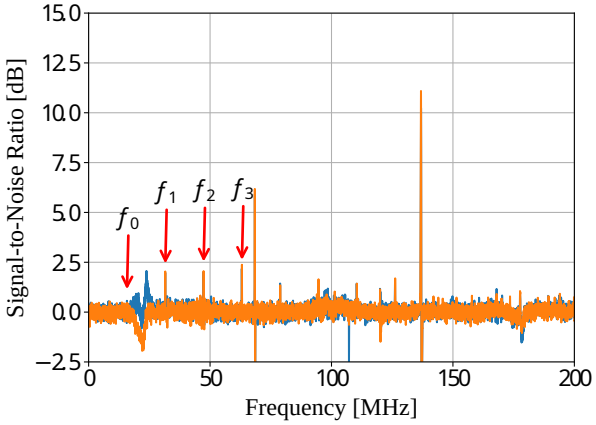

**(a)** Power



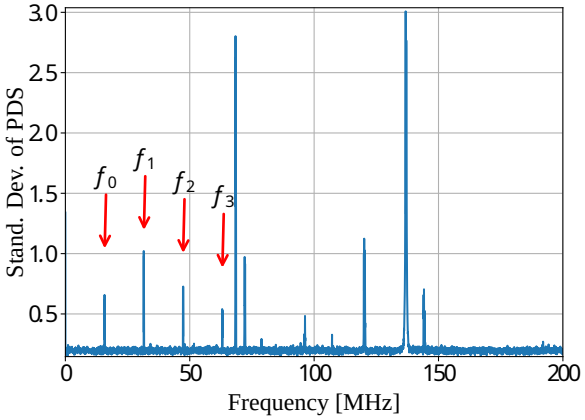

**(b)** EM



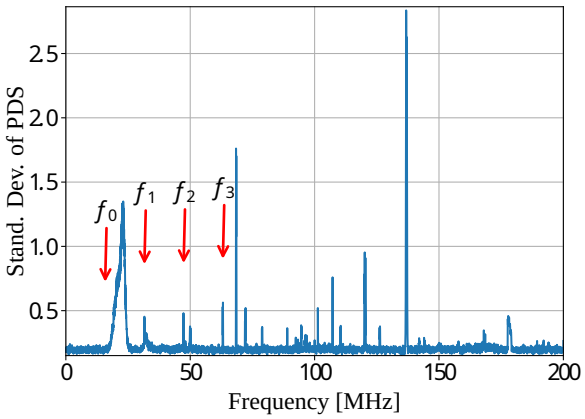

**(c)** Denoised power




**(d)** Denoised EM

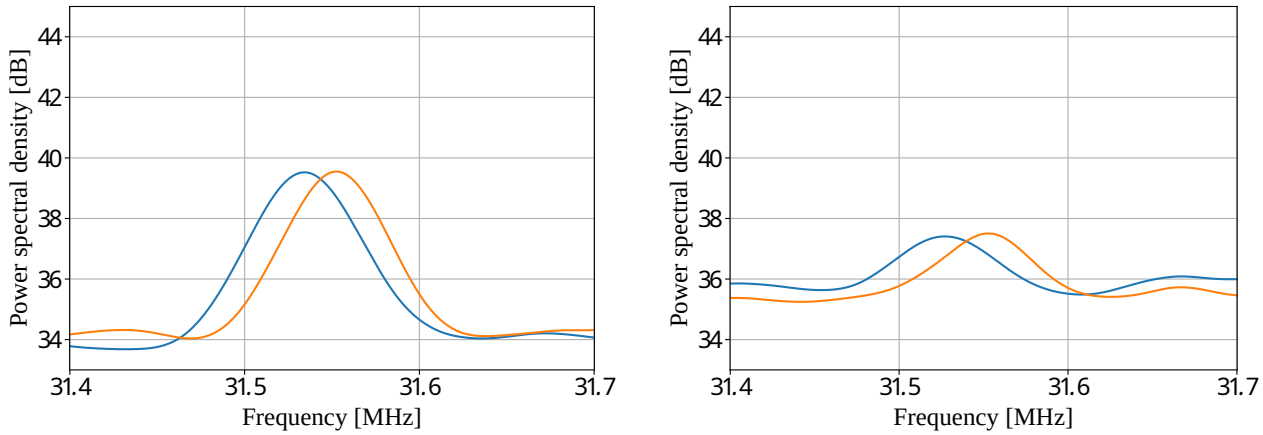



**(e)** Power (Std.)




**(f)** EM (Std.)

**Figure 4.3** Comparison of FoI detection methods for the Loop PUF frequency. (a)-(b): Power Spectral Density (PSD) of exemplary side-channel measurements for $C$ and $\neg C$. (c)-(d): PSD subtracted by PSD from noise measurement. (e)-(f): FoI method using the standard deviation of the PSD among all challenges.




**(a)** Power **(b)** EM

**Figure 4.4** Zoomed PSD for a challenge $C$ (blue) and its complement $\neg C$ (orange).

Fig. 4.5 depicts the match between $\Delta v_C$ and $\Delta \hat{v}_C$. Estimated differences $\Delta \hat{v}_C$ with $\text{sign}(\Delta \hat{v}_C) \neq \text{sign}(\Delta v_C)$ are depicted as filled red squares. Using the method in Eq. (4.2), from 63 Loop PUF bits, only two and three bits result in a wrong guess for the power respectively EM side-channel. For at most $M$ wrong bits the guessing entropy is

$$GE(M) \leq \log_2 \sum_{i=0}^{M} \binom{63}{i}, \tag{4.4}$$

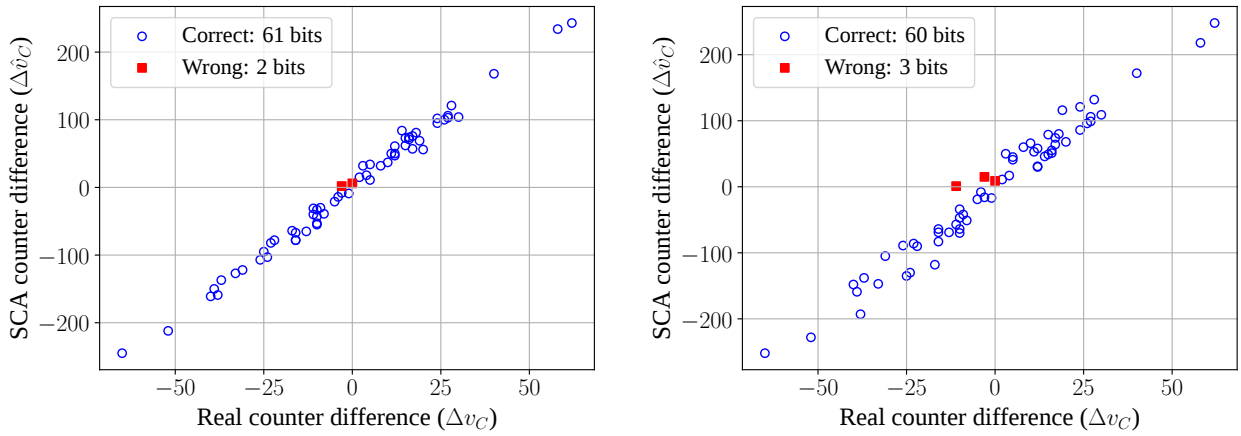
i.e., all possible combinations of $M$ or less erroneous bits have to be considered. Assuming that a guessing entropy of about 32 bits is feasible for brute-forcing, with $M \leq 8$ the 63-bit Loop PUF can be certainly attacked. Notably, in Fig. 4.5 the wrong guesses correspond to smaller frequency differences that are more difficult to resolve. Considering this knowledge, it is possible to improve the search by a smart guessing strategy that first tests bits corresponding to estimated values close to zero. If only the $N_{lim}$ bits with the smallest values of $\Delta \hat{v}_C$ are considered, the guessing entropy is reduced to

$$GE'(M) \leq \log_2 \sum_{i=0}^{M} \binom{N_{lim}}{i}. \tag{4.5}$$

For example, taking $N_{lim} = 41$ of the 63 Loop PUF bits allows for $M \leq 11$ wrong bits to reach a guessing entropy of about 32 bits. Hence, the guessing entropy in Eq. (4.4) is the upper bound if no side-information is considered or in other words more errors $M$ in the attack results can be tolerated by using Eq. (4.5). Note, the smart guessing strategy is only successful if all erroneous bits are contained in the considered $N_{lim}$ bits.

Independent of the search strategy, unstable PUF bits are compensated by an error-correcting step in key generation or even discarded, i.e., an attacker can afford a certain number of wrong bit guesses since also on the device not all 63 bits might be derived correctly. In other words, only for a number of erroneous bits that exceeds the error-correcting capabilities of the device, a strategy as defined by Eqs. (4.4) and (4.5) is required.

Summing up, the response of the Loop PUF can be recovered from non-invasive power and EM measurements using a single measurement per challenge for all but a few unstable bits. Thus, the unprotected Loop PUF design is broken by side-channel attacks.




**(a)** Power **(b)** EM

**Figure 4.5** Results for the SCA of the Loop PUF: Match of real and estimated counter differences from frequency measurements using maxima around 31.55 MHz.

## 4.2 On-Chip Power Analysis of the Loop PUF

Based on the attack from Section 4.1, this section investigates the possibility of on-chip SCA of the Loop PUF primitive and compares the efficiency to classical SCA attacks. Remote power measurements are acquried using a TDC sensor on two Artix-7 FPGAs with different resources to compare differences in the SNR. Further, the relative placement of the targeted PUF and the TDC sensor is varied.

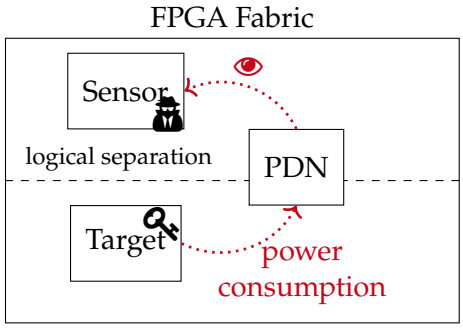



**Figure 4.6** Intra-FPGA remote SCA attack scenario.

In recent years, the possibility to rent computing resources, e.g., on FPGAs, from cloud providers has been emerging. While direct physical access is difficult to achieve for attackers, *remote SCA attacks* constitute a novel attack vector on such devices [88]. These attacks exploit that on shared FPGAs the same Power Distribution Network (PDN) provides the internal voltage $V_{CCINT}$ for all IP cores, even though they are logically separated as depicted in Fig. 4.6. Changes in the power consumption of the target propagate through the PDN, leading to fluctuations of its value $V_{CCINT}$ [89]. An attacker, who shares the same FPGA, can measure the variations by using voltage sensors, e.g., based on ROs [90, 91] or TDCs [92], and conduct SCA from the measurements. Most related work targets implementations of cryptographic algorithms, mainly AES and RSA [88]. Additionally, the inputs of neural network accelerators have been retrieved [93] via remote SCA.

In the domain of shared remote FPGAs, PUFs have been proposed as root of trust in schemes that provide secure computing enclaves on cloud FPGAs [94]. Implementations of oscillation-based PUF and TRNG primitives are available for commercial cloud FPGAs for fingerprinting, authentication and reliability purposes [95]. Furthermore, the use of on-chip sensors has been proposed as low-cost and flexible alternative to classical SCA evaluations for AES [96]. Comparison of remote sensors with

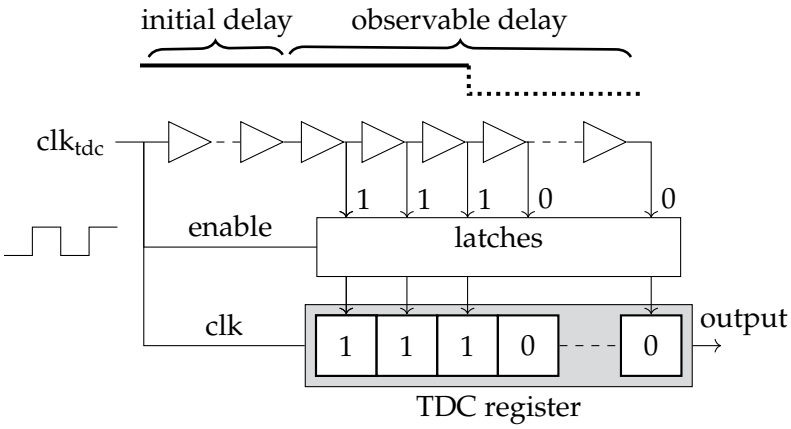



**Figure 4.7** Time-to-Digital Converter (TDC) sensor.

oscilloscope measurements indicate that TDC sensors allow for comparable results for Correlation Power Analysis (CPA) attacks on AES [92, 91].

This section investigates whether remote SCA of the Loop PUF is a possible threat in multi-tenant FPGA scenarios. Additionally, the attack success from on-chip measurements is compared to classical SCA power measurements to provide insights how far low-cost on-chip measurements can be used to evaluate the security of PUFs.

**Attacker Model** In contrast to the other attacks in this thesis, the attacker is *without* physical access to the device under attack, but shares the FPGA with the victim. The attacker has remote access by which she can provide an IP core to the device, that runs in parallel to the victim, and can sent data. The victim uses a PUF to generate a secret key. Algorithms and primitives as well as non-secret parameters used in the victim's IP core are considered public. Concealing this information may add some reverse engineering effort for the attacker, but must not be the basis for security. In order to reveal the secret key generated by the victim's PUF, the attacker uses SCA measurements from a TDC.

### 4.2.1 Time-to-Digital Converter Power Sensor

Fig. 4.7 shows the general concept of a TDC, where a signal propagates through a chain of buffers. The propagation delays of the buffers are sensitive to changes in the supply voltage, i.e., an increased supply voltage decreases the propagation delay and vice versa. Thus, the signal propagation through the delay chain is proportional to the voltage [89]. The voltage sensitivity of the TDC delay can be used as a sensor by tapping the delay line with latches between the buffers; the same signal that is connected to the delay line enables the latches. After a half clock period, the latches are disabled, and the state, i.e., how far the clock signal propagates through the buffer chain, is stored into a register. As the signal passes the buffers sequentially, the register state is a "thermometer" code, i.e., a series of 1's followed by 0's. Other circuits or IP cores that share the same PDN cause a voltage drop in the supply voltage when they are active. Due to the voltage drop, the signal propagation through the TDC delay line is slowed down, and the output register value is reduced. By storing the register values over time, side-channel traces can be generated.

#### Delay Line

The delay line of the TDC is composed of an initial delay and an observable delay: The *initial delay* compensates the propagation delay corresponding to the DC offset of the design. The length of the *observable delay* is designed such that the expected delay variation can be observed. The combination of the clock signal's $clk_{tdc}$ frequency $f_{tdc}$ and the number of buffers of the observable delay determines

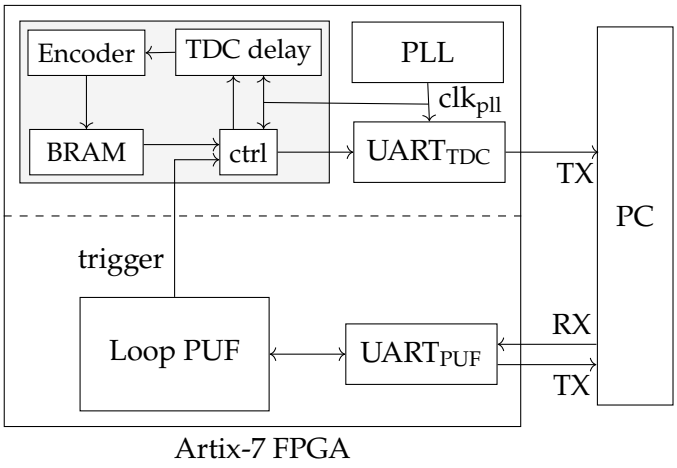



**Figure 4.8** Experimental setup with modules and communication. The dashed line denotes a logical separation.

the sampling rate and resolution of the sensor. These parameters are calibrated before collecting the actual SCA measurements.

#### Bubble Encoder

In order to compress the thermometer code of the register output, a binary encoder is required to convert the TDC output. However, due to uneven propagation delays in the FPGA structure and hold time noise of the latches, so-called *bubbles* may occur at the transition edges, i.e., the last 1 of the output can have one or more preceding 0's [97]. The encoder implementation by Adamič et al. [98], used in the following, leverages LUTs and pipelined adder trees to count the number of 1's in the thermometer code. For at most one or two bubbles in the thermometer code, this is a simple and robust method to correct *bubbles* with little distortion in the resulting sample value.

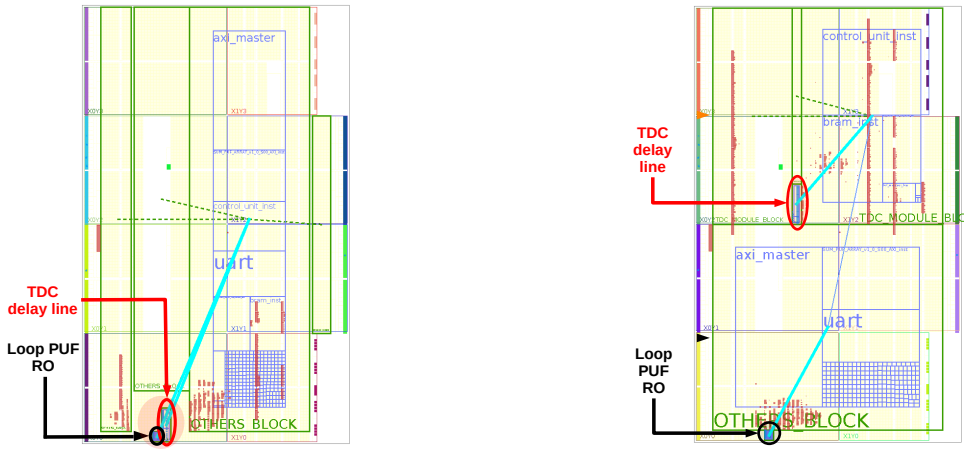
### 4.2.2 Experimental Setup

Fig. 4.8 depicts a block diagram of the experimental setup including the modules of the TDC, the Loop PUF, and the communication between the FPGA and a computer (PC). The modules are logically separated, and the TDC only receives a trigger signal as input and records measurements as soon as the trigger provides a rising edge. Therefore, the TDC module is independent of the attacked module and can be reused in any other target design. Each of the two modules is connected to a separate communication module, using UART modules with a Baud rate of 115200 Baud for the Loop PUF and 460800 Baud for the TDC module.

Similar to Section 4.1.1, a 63-bit Loop PUF is implemented. The control PC sends a challenge to the Loop PUF, which returns the counter value to provide a reference value for the SCA attack. The TDC does not require any control and sends the acquired measurement data to the PC, i.e., a one-directional communication suffices.

The TDC module, in the gray frame, consists of the TDC delay sensor, an encoder, a Block Random Access Memory (BRAM) to store the measurements, and a control unit. Furthermore, a UART sends the acquired data to the PC, and a Phase-Locked Loop (PLL) supplies the TDC and the UART with a separate clock signal. Using a PLL ensures that the TDC module is separated of the system clock, i.e., the sampling rate of the TDC can be adjusted independently.

The encoder converts the sampled TDC values from thermometer code into binary encoding and corrects *bubbles* from the measurements [98]. The resulting intermediate data are stored to BRAM, since the data rate of the communication interface is much slower than the sampling frequency of the sensor. The control unit coordinates data recording, data storage, and data transmission: At the rising edge of the trigger, data is acquired until the BRAM is filled. Then, the data is transmitted over




**(a)** Logical separation, spatial proximity

**(b)** Logical and spatial separation

**Figure 4.9** TDC and Loop PUF placement on the CW305. Edited screenshots from Xilinx Vivado, adapted from [99].

the UART, the BRAM is reset and the next measurement can be taken. As the measurements of the Loop PUF consist of magnitudes more samples than for attacks e.g., on AES, using an Integrated Logic Analyzer (ILA) core [92], which has a limited number of samples that can be transmitted, is not possible. However, a design with intermediate storage and communication interface is closer to a realistic scenario where an ILA is likely out of scope.

The TDC delay path is designed as described in Section 4.2.1, consisting of an initial delay of $N_{init}$ LUTs and $N_{obs.}$ CARRY4 primitives. On the Artix-7 FPGAs, the CARRY4 elements have a smaller delay compared to the LUTs and thus allow for a more fine grained resolution of the observable delay, while for the initial delay a coarse granularity is sufficient to compensate the DC offset. The numbers $N_{init}$ and $N_{obs.}$ have to be calibrated according to the placement, the TDC frequency $f_{tdc}$ and the respective board.

### Target Boards

In order to consider the effect of different SNR due to varying FPGA sizes and a different PDN quality, the experiments are conducted on two FPGA boards featuring distinct Artix-7 FPGAs. Firstly, the *Basys3* development board contains a variety of peripherals for prototyping of digital circuits and features an Artix-7 XC7A35T. Secondly, the *CW305* features an Artix-7 XC7A100T, which has considerably more resources, and is designed for conducting SCA measurements, i.e., it comes with little additional peripherals. The CW305 is also used for a comparison between classical and on-chip SCA measurements of the Loop PUF in Section 4.2.4.

### Placement Configurations

In order to investigate the impact of the distance of the relative placement of the PUF and the sensor module, Fig. 4.9 depicts two scenarios for the CW305 that are investigated. In Fig. 4.9a the Loop PUF and the sensor are logically separated, but spatially adjacent. The TDC delay line is placed close to the oscillator of the Loop PUF. The second scenario in Fig. 4.9b is closer to a real world attack scenario where the FPGA is shared between multiple IPs: the Loop PUF and the TDC sensor are logically and spatially separated. The floorplan is split into two separate blocks by constraints.

**Table 4.1** Overview of experiments with considered FREQUENCIES OF INTEREST (FoIs) ranges determined from counter values.

| | | | FoIs [MHz] | |
|---|---|---|---|---|
| board | $f_{tdc}$ | placement | counter | oscillator |
| CW305 | 40 MHz | close | 7.38 – 7.39 | 14.74 – 14.79 |
| | | separate | 7.38 – 7.39 | 14.72 – 14.78 |
| Basys3 | 30 MHz | close | 6.42 – 6.44 | 12.85 – 12.88 |
| | | separate | 6.44 – 6.45 | 12.85 – 12.89 |
| CW305 | 16 MHz | close | 7.46 – 7.47 | – |

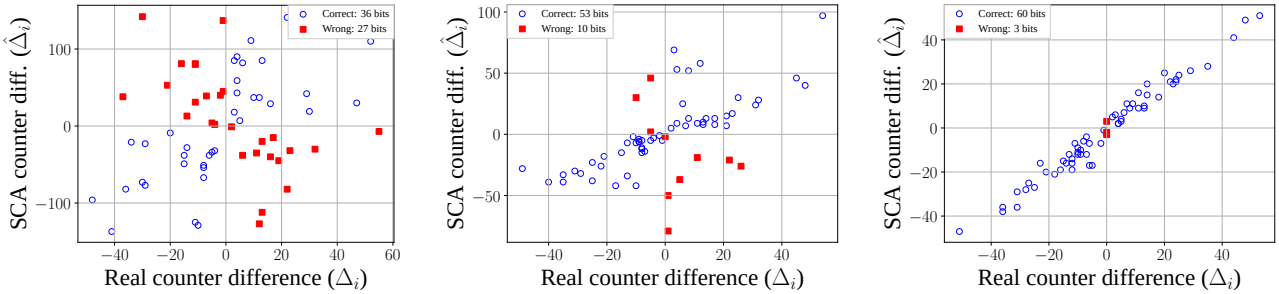
### 4.2.3 Practical Results of On-Chip Power Analysis

For all results of the TDC-based SCA of the Loop PUF, the Loop PUF is activated for $n_{acq} = 2^{19}$ cycles of a $f_{ref} = 100\,\text{MHz}$ clock ($T_{ref} = 10\,\text{ns}$) corresponding to around 5.2 ms for each challenge. Time domain measurements are acquired by the TDC sensor (or an oscilloscope for classical SCA measurement) and transformed into the frequency domain. In order to improve the SNR, multiple measurements, also denoted as repetitions in the following, are acquired per challenge. Subsequently, the frequency spectra of several repetitions are averaged. From the averaged frequency spectra, peak detection is used to get the estimated oscillation frequencies. In order to facilitate the analysis, the FoI range, in which the peak detection is applied, is limited to the range of frequencies calculated from the actual counter values by leveraging Eq. (4.1). An example spectrum and the zoom to the FoI range are provided in Appendix B.2. In a real attack scenario, the attacker could narrow the FoI based on a comparison of the spectrum during activity of the PUF compared to normal operating conditions, e.g., by one of the methods presented in Section 4.1.3.

Table 4.1 provides an overview of the different experimental scenarios and the corresponding FoIs determined from the counter values. As expected for a PUF, the Loop PUF frequency is board dependent, due to differing available resources used. In addition, it differs slightly for different measurement campaigns due to slightly varying environmental conditions and other noise effects. Note that the oscillator of the Loop PUF and the counter in Fig. 2.5a are separated by a TOGGLE FLIP-FLOP (T-FF), which acts as a frequency divider. From the device's perspective, the counter value depicts the halved frequency of the Loop PUF, but does not affect the sign of the bit derivation. From the attacker's perspective, two possible FoIs have to be considered: due to the frequency-dividing T-FF, the feedback buffer chain oscillates at about twice the frequency as the counter.

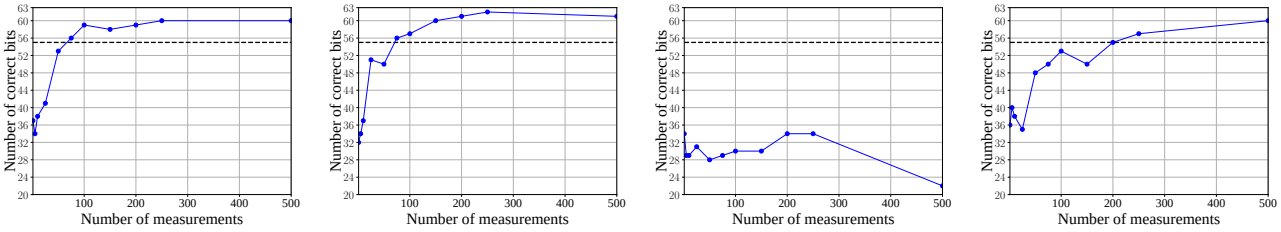
According to the Shannon-Nyquist theorem, the frequency $f_{tdc}$ of the TDC has to be at least twice the frequency of the Loop PUF. For the different scenarios in Table 4.1, the condition is fulfilled with an additional margin. The BRAM in the TDC sensor module must be able to store $T_{ref} \cdot n_{acq} / f_{tdc}$ samples. Consequently, the maximum sampling frequency $f_{tdc}$ of the TDC is limited by the available BRAM blocks of the FPGA. Finally, while a high sampling rate $f_{tdc}$ is desirable to increase the frequency range and resolution, a lower $f_{tdc}$ can increase the sensitivity as the signal can propagate further into the observable part of the delay chain. In the first four rows in Table 4.1 a sampling rate $f_{tdc}$ is chosen that allows for resolving the FoIs of counter and oscillator, and increases the frequency resolution. On the other hand, in the last row in Table 4.1 the sampling rate $f_{tdc}$ is chosen close to the minimum possible value that resolves the counter FoIs. This increases the sensitivity of the TDC sensor, but comes at the cost of not being able to resolve the oscillator FoIs.

To evaluate the attack success, the signs of the real counter difference and the counter difference retrieved based on the SCA frequency peaks are compared, similar to Section 4.1.4. Examples of the match for the CW305 with $clk_{tdc} = 40\,\text{MHz}$ for different numbers of repetitions are depicted in Fig. 4.10. With an increasing number of repetitions, the SNR is increased from averaging and the match becomes more precise. Assuming that a guessing entropy of about 32 bits is feasible for brute-




**(a)** 1 repetitions **(b)** 100 repetitions **(c)** 500 repetitions

**Figure 4.10** Comparison of real and SCA counter differences from oscillator FoIs on the CW305 for $clk_{tdc}$ = 40 MHz and separate placement of PUF and sensor.

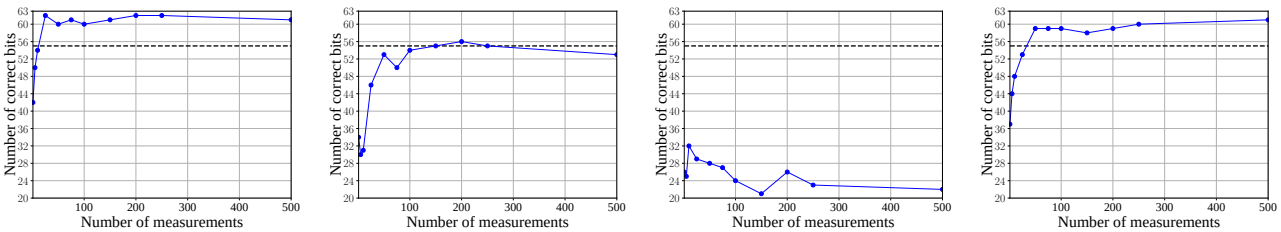



**(a)** Close, counter FoI **(b)** Close, oscillator FoI **(c)** Separate, counter FoI **(d)** Separate, oscillator FoI

**Figure 4.11** Attack results on CW305 with $clk_{tdc}$ = 40 MHz.

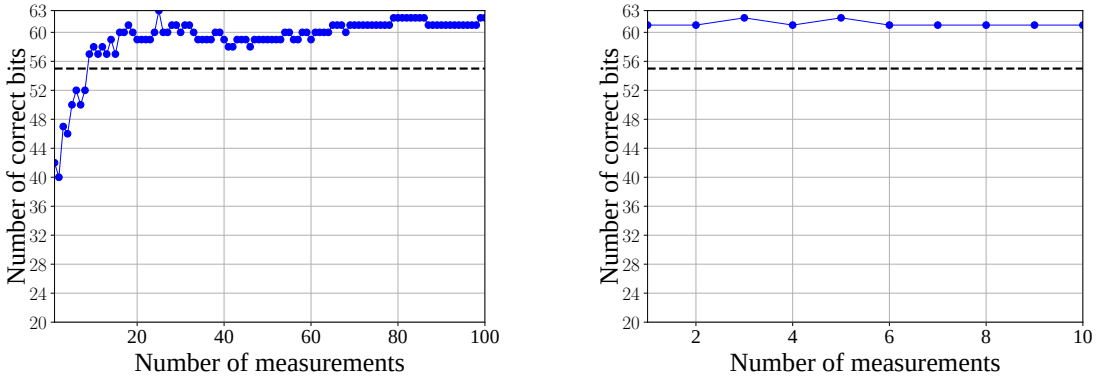
forcing, the Loop PUF can be attacked if the attack has $M \leq 8$ erroneous bits according to Eq. (4.4). Similar to the classical SCA results in Section 4.1.4, for a sufficiently low SNR most wrong guesses occur for frequency differences close to zero as highlighted by the red filled squares in Fig. 4.10c and Appendix B.1. Therefore, it would be possible to improve the search by a smart guessing strategy that first tests the values close to zero and leads to a reduced guessing entropy according to Eq. (4.5). However, as the brute-force strategy leads to successful results, it is adopted for the following results.

Figs. 4.11 and 4.12 depict the SCA results on the CW305 and the Basys3 boards as outlined in the first four rows of Table 4.1. The dashed lines denote the brute-force limit of eight erroneous bits. The plots show the number of correctly retrieved bits from the 63-bit Loop PUF with increasing number of measurements used for averaging the spectra. In general, averaging over multiple repetitions allows for aggregating information, whereas a single measurement would not suffice for a successful attack. The number of recovered bits reaches a saturation point, but in some cases decreases slightly afterwards. The variations can be attributed to changing environmental conditions as the entire measurement campaigns are taken over more than 30 hours. On both boards, depending on the placement and the targeted FoI range, the on-chip measurements from the TDC sensor allow for reducing the remaining uncertainty about the PUF response down to a few bits.




**(a)** Close, counter FoI **(b)** Close, oscillator FoI **(c)** Separate, counter FoI **(d)** Separate, oscillator FoI

**Figure 4.12** Attack results on Basys3 with $clk_{tdc}$ = 30 MHz.




**(a)** On-chip $f_{tdc}$ = 16 MHz

**(b)** Classical $f_s$ = 156.25 MS/s

**Figure 4.13** Comparison of attack results on CW305 for (a) best case on-chip and (b) classical power SCA measurements.

**Influence of the Placement** In Figs. 4.11a, 4.11b, 4.12a and 4.12b for a close placement of the TDC sensor and the Loop PUF, the FoIs from the counter can be more easily exploited than the ones from the oscillator, i.e., less measurements are required to retrieve a high number of correct bits. On the other hand, in Figs. 4.11c, 4.11d, 4.12c and 4.12d for a separate placement, only the FoIs corresponding to the oscillator allow for attacking the Loop PUF.

A possible explanation, why attacks on the counter FoIs are not possible for separate placement, is that the oscillator has a power consumption that affects the PDN stronger and thus can be sensed independent of the sensor placement. For close placement, the sensor also captures the counter's local influence to the PDN and can resolve the corresponding FoIs better, which leads to an increased number of correctly recovered bits.

**Influence of FPGA Size and TDC Sampling Frequency** Except for the close placement and oscillator FoIs in Figs. 4.11b and 4.12b, the results for the Basys3 board show a faster increase of the number of correct bits. The measurements on the Basys3 board use a lower sampling frequency of $f_{tdc}$ = 30 MHz, which increases the sensitivity, affecting the attack results positively. As the overall results between Figs. 4.11 and 4.12 are in the same range, the SNR seems to play a negligible role. In order to investigate the effect of the sampling frequency further, Fig. 4.13a depicts the attack success on the CW305 with $f_{tdc}$ = 16 MHz. Due to the low sampling rate only the counter FoIs can be targeted as for the oscillator FoIs the sampling rate is too low. Compared to Fig. 4.11a, where with 100 traces five erroneous bits remain, in Fig. 4.13a with 18 traces only three erroneous bits remain, with 25 traces all bits are retrieved correctly. In conclusion, to increase the attack success a low TDC sampling rate is beneficial.

### 4.2.4 Comparison with Classical Side-Channel Analysis

Finally, to compare the attack success of the on-chip power measurements to measurement from a classical SCA setup, power measurements from the CW305 are acquired with a sampling rate of $f_s$ = 156.25 MS/s. Similar to Section 4.1.4, peak detection is applied to the FoI range from 31.73 MHz to 31.95 MHz, corresponding to the first harmonic of the oscillator. From Fig. 4.13b, with a single trace only two bits are not recovered correctly resulting in a successful attack. In the best case on-chip scenario with $f_{tdc}$ = 16 MHz in Fig. 4.13a, nine repetitions are required for feasible brute-force search and at least 18 repetitions to achieve the same precision as classical SCA.

The comparison in Fig. 4.13 shows that classical SCA is – as it might be expected – indeed more powerful when compared to on-chip SCA. However, the results also highlight that on-chip measurements are capable of attacking the Loop PUF and are a possible attack vector in shared FPGAs. The

limiting factors for the considered remote attack are the TDC sampling rate $f_{tdc}$ and the transmission of the acquired data. The relatively low oscillation frequency of the Loop PUF allows for setting a low sampling rate of the TDC that increases the resolution and reduces the acquired data at the same time. For oscillators with a higher frequency, e.g., ROs with less delay elements, an increased sampling frequency would be needed to capture the FoIs. In order to maintain a sufficient resolution, the TDC could be modified to use parallel delay lines [100]. Additionally, the amount of data could be reduced at the cost of additional sensor complexity, e.g., by calculating the frequency transformation on-chip and transmitting only the relevant frequency bins. Some a priori knowledge on the attacker's side in terms of the FoIs would be required for this approach.

From the results, similar to quick evaluation of AES from on-chip measurement [96], SCA evaluations of PUFs without expensive measurement equipment are in principle possible. Due to the long time needed for data transmission, the application seems to be limited to scenarios, where oscilloscope-based testing it not possible or comes with a large overhead.

## 4.3 Securing the Loop PUF

To thwart SCA of the Loop PUF, a masking countermeasure is proposed in this section. First, the general concept of the temporal masking scheme is introduced in Section 4.3.1. Second, limitations and constraints of the attack are discussed in Section 4.3.2, which are leveraged to make the Loop PUF self-secured by using the counter LSB as a random bit in Section 4.3.3. Finally, Sections 4.3.4 and 4.3.5 evaluate the mask quality and provide results for SCA for the proposed countermeasure.

### 4.3.1 Temporal Masking

The measurements of the Loop PUF to derive a PUF bit $r_C$ are performed sequentially, i.e., the measurement for challenge $C$ is followed by the measurement for its complement $\neg C$. The ordered sequential measurements are exploited by the SCA in Section 4.1.4. However, reordering of measurements does not affect PUF quality metrics as it has no effect on the oscillation frequency. Thus, to protect the sequential measurements against SCA, the order of measurements to derive a PUF response bit can be randomized. In Algorithm 2, the bit derivation for $r_C$ is randomized by a 1-bit mask $m$, which is unknown and unpredictable for an attacker. As the countermeasure modifies the temporal order of the applied challenges, it is denoted as *temporal masking* in the following.

**Algorithm 2** Temporal Masking: Protected Loop PUF Operation

**Input:** Challenge $C$
**Input:** Measurement time in terms of periods $n_{acq}$ of the reference clock
**Input:** mask $m$ (1-bit random variable)
**Output:** Response $\delta_C$ (a signed integer whose sign is mapped to the secret bit $r_C$)
1: Set current challenge $C' = m \; ? \; C : \neg C$
2: Count oscillations of the Loop PUF for $n_{acq}$ cycles of the reference clock $\Rightarrow v_{C'}$
3: Set current challenge $\neg C'$
4: Count oscillations of the Loop PUF for $n_{acq}$ cycles of the reference clock $\Rightarrow v_{\neg C'}$
5: Compute $\delta_C = m \; ? \; v_{C'} - v_{\neg C'} : v_{\neg C'} - v_{C'}$
6: **return** $\delta_C$ with $r_C = \mathrm{MSB}(\delta_C) \in \{0, 1\}$

Comparing the operation mode of the Loop PUF with temporal masking in Algorithm 2 to the basic operation mode in Algorithm 1, the mask bit $m$ determines whether $C$ or $\neg C$ is applied first (Lines 1, 3): If $m$ is logically 0, the sequence of challenges is $C \prec \neg C$; Otherwise, if $m$ is logically 1, the order is $\neg C \prec C$. Consequently, for an unknown mask bit $m$ the attacker cannot determine the order of frequency measurements, which impedes the SCA from Section 4.1.4.

The order of the measurements determines the secret bit according to Algorithm 1, i.e., without further modification, a changed order of measurements leads to a wrong sign derived from the frequency difference. Therefore, the sign is corrected by considering the order of measurement also in the subtraction (Line 5). The mask bit $m$ determines the order in which the frequencies are subtracted such that the final result is independent of $m$ but still cannot be observed by an attacker.

### 4.3.2 Towards a Self-Secured Loop PUF: Understanding SCA Limitations and Constraints

In order to understand general limitations of the SCA presented in Section 4.1.4, this section revisits the fundamentals of the attack and provides constraints regarding the possible frequency resolution of observations. Understanding the resulting limitations of the SCA approach enables the countermeasures proposed in Section 4.3.3.

In Section 4.1.4, the attacker acquires time domain signals of length $T_{sca}$ with an oscilloscope at a sampling frequency $f_s$, which are subsequently transformed into the frequency domain. First, the general case is considered, where the attacker measures for a time $T_{sca} \leq T_{acq}$, i.e., at most the time for which the Loop PUF is active, to verify that $T_{sca} = T_{acq}$ is the best setup from an attacker's perspective. The smallest frequency $f_{min}$, which can be resolved, is the frequency, for which exactly one complete period of the oscillation fits into the observation window $T_{sca}$. In case of the Loop PUF, the maximum observation time is the acquisition time $T_{acq}$, i.e.,

$$f_{min} := \frac{1}{T_{sca}} \geq \frac{1}{T_{acq}} = \frac{f_{clk}}{n_{acq}}. \tag{4.6}$$

The maximum frequency $f_{max}$ that can be resolved, is determined by the Shannon-Nyquist sampling theorem as $f_{max} = f_s/2$ for the sampling frequency $f_s$. Thus, the observable frequency range[1] is bounded to

$$\frac{1}{T_{acq}} \leq f_{min} \leq f \leq f_{max} = \frac{f_s}{2}. \tag{4.7}$$

For a measurement period of $T_{sca}$, the number of sampling points, i.e., the length of the applied FFT is

$$N_{FFT} = f_s \cdot T_{sca}. \tag{4.8}$$

Note that for real valued time domain signals the spectrum is symmetric. Therefore, an FFT of length $N_{FFT}$ maps the signal into $N_{FFT}/2 + 1$ frequency bins ranging from DC to $f_{max}$. The frequency resolution of the FFT frequency bins is

$$\Delta_{FFT} = \frac{f_{max}}{N_{FFT}/2} = \frac{f_s}{N_{FFT}} = \frac{1}{T_{sca}} \geq \frac{1}{T_{acq}}. \tag{4.9}$$

In other words, a longer measurement time $T_{sca}$ allows the attacker to obtain a better resolution of the frequency differences. An attacker is expected to get the best result if the entire acquisition time $T_{sca} = T_{acq}$ is measured as in Section 4.1.4. It is worth mentioning that the frequency resolution is independent of $f_s$ and only determined by the design parameter $T_{acq}$, i.e., it cannot be improved by the attacker.

The FFT induces a certain quantization error, i.e., the observed bin center frequency $\hat{f}$ represents a value range for the real oscillation frequency $f_{real}$ of the Loop PUF. From Eq. (4.9), $f_{real}$ is bounded by the width of the frequency bins to

$$\hat{f} - \frac{1}{2 \cdot T_{acq}} \leq f_{real} \leq \hat{f} + \frac{1}{2 \cdot T_{acq}}. \tag{4.10}$$

[1] Note that technically, the smallest frequency that can be resolved is 0 Hz, i.e., the DC component. However, in Eq. (4.7) we are concerned with the *observable* frequencies.

Assuming all frequencies within a specific bin appear with the same probability, the best guess an attacker can make for the counter value according to Eq. (4.3) from the observed $\hat{f}$ is therefore

$$\hat{v}_C = \left\lfloor \left( \hat{f} \pm \frac{1}{2 \cdot T_{acq}} \right) \cdot T_{acq} \right\rfloor = \left\lfloor \hat{f} \cdot T_{acq} \right\rfloor \pm 1. \tag{4.11}$$

Regarding limitations and constraints of the SCA and implications for possible countermeasures, from Eqs. (4.10) and (4.11) the following holds:

1. If the frequency difference of two challenges $C$ and $\neg C$ is $|f_C - f_{\neg C}| > \Delta_{FFT}$, the resulting PUF response bit $r_C$ is always revealed by an attack.

2. If $|f_C - f_{\neg C}| \leq \Delta_{FFT}$, the probability that both $f_C$ and $f_{\neg C}$ are in the same FFT bin, i.e., indistinguishable for an attacker, increases with decreasing distance of the frequencies. The attack will succeed for small frequency differences only with a certain probability.

3. While the sign of the counter difference can be revealed, an attacker will fail in deriving the LSB of the counters.

Note that regarding Item 2, intentionally designing a Loop PUF with closeby frequencies does not serve as a countermeasure: The comparison of frequencies close to each other is not desirable from a PUF perspective, because bits derived from such a comparison are less robust against noise. The conclusions in Items 1 and 2 emphasize the necessity for countermeasures to protect the Loop PUF. Additionally, Item 3 substantiates that the LSB of a counter cannot be revealed by the attack. Consequently, the LSB is used in Section 4.3.3 as a random bit to protect the Loop PUF.

### 4.3.3 Self-Secured Loop PUF Using Randomness From the Counter LSB

The temporal masking scheme from Section 4.3.1 allows for masking the Loop PUF at the expense of a random bit per PUF bit. This section addresses the question how to efficiently implement the masking scheme without the effort of an additional Random Number Generator (RNG). Based on the limitations of the SCA frequency resolution derived in Section 4.3.2, the LSB of the Loop PUF counter is suggested as 1-bit RNG, resulting in a *self-secured* Loop PUF.

Algorithm 3 describes the response generation of the self-secured Loop PUF with temporal masking and an LSB-based mask. The algorithm takes the acquisition time in terms of the number of clock cycles $n_{acq}$ of a reference clock as an input during design time. When executed, all Hadamard codewords except of the all-zero challenge $C_0$ (Line 1) are derived. Note that the Hadamard codewords can be computed during runtime and do not require additional memory. The successive codeword can be computed parallel to applying the current codeword to the PUF.

Following Sections 2.2.2 and 4.1.1, the all-zero challenge $C_0$ cannot be used to extract a PUF bit as it is biased if the delay stage is imbalanced. However, it can be used to derive a mask bit (Lines 2 to 4) for the generation of the first response bit. The oscillations of the Loop PUF for $C_0$ are measured and the LSB of the resulting counter value is taken as mask bit $m$.

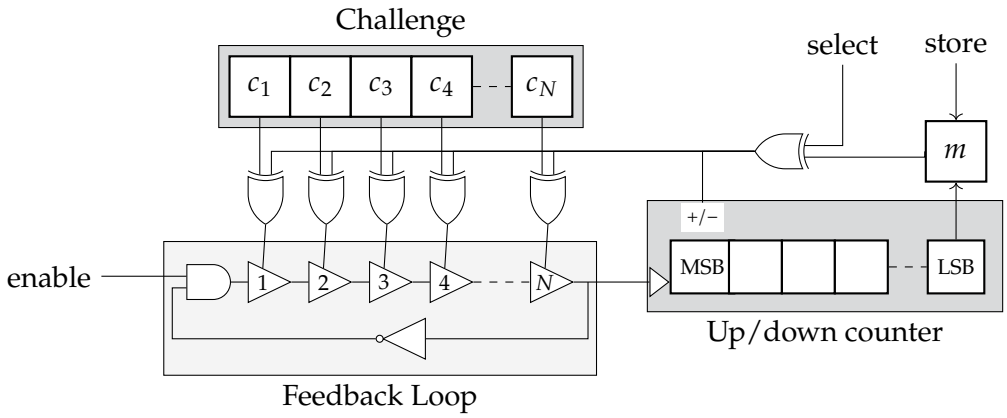



**Figure 4.14** Schematic of the protected Loop PUF structure.

**Algorithm 3** Protected Loop PUF

**Input:** Measurement time in terms of periods $n_{acq}$ of the reference clock
**Output:** $N-1$-bit PUF response $\mathbf{r} = [r_{N-1}, \ldots, r_1]$

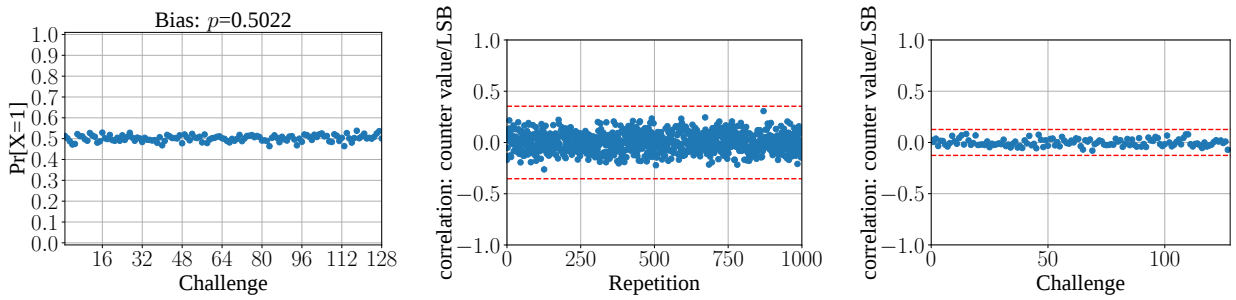
1: Compute the Hadamard codewords set $C = \{C_1, \ldots, C_{N-1}\}$ with $\mathrm{HW}(C_i) = N/2$
2: Set current challenge $C' = C_0 = [0 \ldots 0]$
3: Count oscillations of the Loop PUF for $n_{acq}$ cycles of the reference clock $\Rightarrow v_{C'}$
4: Set mask $m = \mathrm{LSB}(v_{C'}) \in \{0, 1\}$
5: **for all** $i = N-1$ down to 1 **do**
6: Set current challenge $C' = m \; ? \; C_i : \neg C_i$
7: Count oscillations of the Loop PUF for $n_{acq}$ cycles of the reference clock $\Rightarrow v_{C'_i}$
8: Set current challenge $\neg C'_i$
9: Count oscillations of the Loop PUF for $n_{acq}$ cycles of the reference clock $\Rightarrow v_{\neg C'_i}$
10: Compute $\delta_C = m \; ? \; v_{C'_i} - v_{\neg C'_i} : v_{\neg C'_i} - v_{C'_i}$
11: Set $r_i = \mathrm{MSB}(\delta_C) \in \{0, 1\}$
12: Set mask $m = \mathrm{LSB}(v_{C'_i}) \in \{0, 1\}$
13: **end for**

Subsequently, all other $i = 1, \ldots, N-1$ Hadamard codewords $C_i$ and their complements $\neg C_i$ are applied to the Loop PUF. The measurement order of $C_i$ and $\neg C_i$ is randomized by the current mask bit $m$ (Line 6 to 10) according to the steps from Algorithm 2.. A secret PUF bit $r_i$ is derived from the MSB of the counter difference. Finally, the mask bit is updated to the random LSB of the counter value $v_{C'_i}$ protecting the next measurement.

Fig. 4.14 sketches a possible hardware implementation of the self-secured Loop PUF omitting generation of the Hadamard codewords, reference counter, state machine, output registers, and reset tree. In an actual design, the state machine would cause generation of Hadamard codewords and loading of codewords to the challenge register while resetting the counter. An up/down counter might be used for counting the periods of the Loop PUF.

Starting with the all-zero challenge, $m = 0$ and *select* $= 0$, the number of Loop PUF oscillations within the acquisition time are measured. Without loss of generality, it can be assumed that the counter is counting upwards in this mode. Setting *store* $= 1$ for one cycle after $n_{acq}$ clock cycles, the LSB of the resulting counter value is buffered as the first mask bit.

Subsequently, four main states are repeated until all $N-1$ challenges have been applied to the Loop PUF: *(i)* The mask bit from the buffer is applied to the input of the XOR tree, another challenge is loaded, and the counter is reset. *(ii)* The *select* signal in the design is set to logical 0 and *enable* is set to logical 1. *(iii)* After $n_{acq}$ cycles of the reference clock, the LSB is buffered but not yet used as $m$,




**(a)** LSB bias

**(b)** Correlation for different challenges

**(c)** Correlation for different measurements

**Figure 4.15** Empirical analysis of the LSB-mask: (a) Relative frequency of the LSB for all Hadamard challenges. Correlation between LSB and frequency over (b) multiple challenges, (c) repeated measurements.

*select* is switched to logical 1. *(iv)* After another $n_{acq}$ cycles of the reference clock, the MSB is taken as a secret bit.

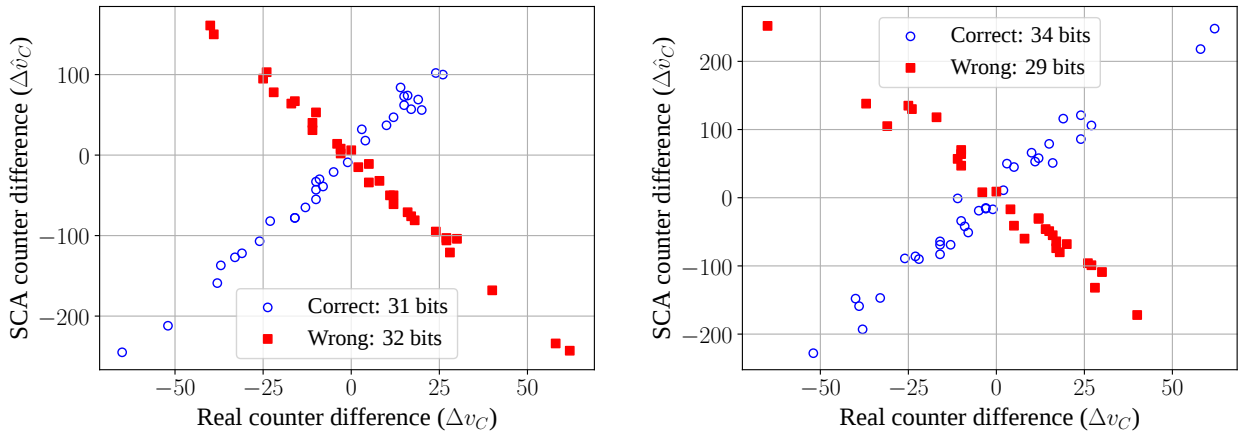
The structure of the design causes that if $m \oplus select = 0$, the counter counts upwards and $C$ is applied to the PUF. If $m \oplus select = 1$, $\neg C$ is used while counting downwards. In other words, if $m = 0$, first $C$ is applied while counting upwards before $\neg C$ is applied while counting downwards; if $m = 1$ the order of $C$ and $\neg C$ as well as the counting direction in state *(ii)* and *(iii)* are reversed, so that after the complete sequence of states the up/down counter always contains the correct frequency difference and no inversion of the MSB is required.

### 4.3.4 Empirical Analysis of the LSB-Mask

Temporal masking is effective, if the attacker cannot predict the mask bit $m$. Section 4.1.4 shows that the LSB is not resolvable by the suggested measurement strategy. Hence, the question remains if the attacker can predict the LSB by some other means, e.g., if the LSB *(i)* has an exploitable bias or *(ii)* is correlated to the oscillation frequency.

**Bias** A bias is considered exploitable if LSB($v$) for the same challenge is equal for all devices or if LSB($v$) exhibits a global bias with respect to all challenges. The former case is excluded from further analysis, since a bias over devices implies the same frequencies for a challenge over all devices. Consequently the PUF quality would be low and some redesign is required. To rule out a bias on the device that influences the quality of the mask bit, Fig. 4.15a depicts the relative frequencies of the LSB for all challenges, where each challenge is measured 1000 times, and no apparent bias exists among challenges. The global bias of all LSBs from all challenges is 0.5022, which is within the expected range. Note that in Fig. 4.15a the LSBs of the counter values from all $2 \times N = 128$ challenges are included in the analysis, while for the protection with the temporal masking scheme only $N - 1 = 63$ random bits are needed. However, according to Algorithm 3 the challenges used to derive the random bits are not known a priori, but depend on previous random bits. Therefore, all 128 counter LSBs are analyzed.

**Correlations With Frequency** Regarding correlations to the oscillation frequency, two cases are considered: First, the attacker might take advantage from correlations between the LSB and the frequency over multiple repeated measurements for a fixed challenge. This would indicate that a certain guessed frequency corresponds to a certain LSB. Second, the attacker might take advantage of a correlation between the LSB and the frequency over multiple challenges, which would indicate a general dependency between frequency and the LSB. Figs. 4.15b and 4.15c refute the existence of both kinds of correlations in the design. Both figures show the respective correlation values between frequency and LSB along with a threshold depicted in dashed red. Values below the threshold, given




**(a)** Power **(b)** EM

**Figure 4.16** Attack results from SCA of the self-secured Loop PUF: Match of real counter differences and estimated counter differences.

by $\pm 4/\sqrt{N}$, are not significantly different from zero with a confidence of 99.99% [51]. The number of observations $N$ used to calculate the correlation is $N = 128$, i.e., the number of different challenges, and the experiment is repeated for 1000 measurements in Fig. 4.15b. In Fig. 4.15c, $N = 1000$ different measurements are correlated and the experiment is repeated for each challenge. In neither case is the significance threshold exceeded indicating no correlations between LSB and frequency.

To sum up, the LSB of the Loop PUF counter is suited as a mask bit. It does not show significant bias, nor is the LSB correlated to the frequency of the oscillation, which an attacker could observe. Establishing these properties makes the self-secured Loop PUF a low-complexity and secure design.

### 4.3.5 Side-Channel Analysis of the Self-Secured Loop PUF

Finally, the effectiveness of the self-secured PUF design on practical measurements is evaluated. In order to assure a fair comparison, the exact same measurements as in Section 4.1.4 are used, but the order of measurements for $C$ and $\neg C$ presented to the attacker is modified according to Algorithm 3. The random bit $m$ is determined from the counter values obtained from the device. In Fig. 4.16, the attacker's capability to estimate the counter difference is depicted. Note that the attacker tries to guess the MSB as well as the LSB. From the remarks from Section 4.3.2 it is evident that the LSB cannot be retrieved, which is reflected in Fig. 4.16: Due to the randomized acquisition order, the relationship between real counter differences and SCA-based counter difference estimates is broken and the self-secured Loop PUF is effectively hardened against SCA.

## 4.4 Remarks on the Temporal Masking Countermeasure

The temporal masking scheme allows for protecting the Loop PUF against SCA by randomizing the observations for an attacker. This section elaborates on the required quality of the masks in Section 4.4.1, and the impact of the measurement time in Section 4.4.2.

### 4.4.1 Required Mask Quality

The empirical analysis of the LSB in Section 4.3.4 shows that the design provides mask bits that are uniformly distributed. However, it is worth noting that temporal masking still provides protection for biased mask bits as long as the bias per bit is unknown to the attacker. Considering LSB-based masks, the attacker can count the occurrences of the frequencies $\hat{f}_C$ and $\hat{f}_{\neg C}$ for each challenge $C$ from

repeated observations of the PUF response generation. This allows for determining the deviation $\delta_{bias}$ of the relative frequency from its ideal value of 0.5. However, the attacker is not able to assign $\hat{f}_C$ and $\hat{f}_{\neg C}$ to individual observations. Only the absolute value $|\delta_{bias}|$ of the deviation is revealed. Thus, the estimated relative frequency of the mask $m$ is $0.5 \pm |\delta_{bias}|$, which still provides one bit of uncertainty as the sign cannot be determined by the attacker.

Note that biases on different devices have to be different. Otherwise, the attacker could conclude about the bias from a template device to the attacked device. Furthermore, for biased mask bits, the bias for different bits on the same device must not be correlated. For example, if all bits are biased in the same direction, the uncertainty is reduced to one bit for the entire PUF response and temporal masking is circumvented.

### 4.4.2 Impact of Measurement Time

The frequency measurement depends largely on the measurement window $T_{acq}$. From Section 4.3.2, the attack becomes more difficult with the reduction of $T_{acq}$, as the FFT accuracy decreases. Thus, a naïve countermeasure would be the reduction of the measurement time. Additionally, the latency is proportional to $T_{acq}$ making a design with smaller $T_{acq}$ more efficient. However, a small $T_{acq}$ significantly reduces the reliability because the quantization noise of the counting process is increased. Hence, the best compromise depends on different factors such as the required latency and reliability of the key generation.

Neglecting latency, a large $T_{acq}$ provides a higher reliability of the PUF response bits. As a larger $T_{acq}$ comes at the cost of side-channel leakage, a countermeasure like temporal masking is inevitable. Yet, temporal masking provides security benefits independent of the measurement time, since it impedes attacks independent of the capability of the attacker to resolve frequencies. It is, e.g., still effective against fault attacks where an attacker is able to extend the measurement time by decreasing the frequency of the reference counter.

# 5 Side-Channel Analysis and Protection of the Two-Metric Helper Data Scheme

This chapter analyzes the peculiarities regarding SCA and its mitigation for bit derivation schemes other than the sign-based bit derivation treated in the previous chapters. Section 5.1 provides several insights regarding attack vectors of amplitude-based derivation schemes resulting in requirements for a protection of the Loop PUF independent of the bit derivation.

The remainder of the chapter focuses on the Two-Metric Helper Data (TMHD) scheme presented in Section 2.3 combined with the Loop PUF. First, Section 5.2 proves that the temporal masking countermeasure introduced in Section 4.3.1 does not provide protection for amplitude-based bit derivation. A novel attack against the TMHD scheme is shown, which makes use of the *amplitude* – in contrast to the *sign* – of the frequency differences from ring-based PUFs. The feasibility of the attack is shown for a 63-bit Loop PUF. Second, Section 5.3 takes a first step into the direction of protecting the TMHD by combining a modification of the helper data metrics with the randomization of the challenge order, denoted as *Challenge Randomization* countermeasure. It turns out that lightweight generation of the required randomness can be broken, and TRNG-based implementations are costly with a non-deterministic runtime. Therefore, Section 5.5 presents the Interleaved Challenge Loop PUF (ICLooPUF), which is a design modification of the Loop PUF that hides side-channel leakage in the first place. A theoretical analysis of the protection is provided as well as practical results that show the effectiveness of the new protected PUF primitive.

The results in this chapter are based on the publications *Tebelmann et al.: "Analysis and Protection of the Two-Metric Helper Data Scheme" in* Constructive Side-Channel Analysis and Secure Design, *Springer International Publishing, 2021, pp. 279-302* [101] and *Tebelmann/Danger/Pehl: "Interleaved Challenge Loop PUF: A Highly Side-Channel Protected Oscillator-Based PUF" in* IEEE Transactions on Circuits and Systems I: Regular Papers, *2022, vol. 69, no. 12, pp. 5121-5134* (©2022 IEEE) [21].

## 5.1 Requirements for a Universal Protection of the Loop PUF

The temporal masking countermeasure from Section 4.3.1 is designed to protect the sign-based bit derivation from Loop PUF counter differences. However, as will be shown in Section 5.2 for the TMHD scheme, amplitude-based bit derivation is not protected by the temporal masking countermeasure. In addition to the attack on the TMHD scheme, other amplitude-based bit derivation schemes from Section 2.3.3 are impacted by SCA that reveals information about the amplitude of a frequency difference as well. This section first outlines possible threats to the EPQ, and the order encoding bit derivation schemes. Subsequently, bit derivation methods from Section 2.3 are compared regarding the secret to be protected. This allows for determining the requirements for a universal protection of the Loop PUF that applies to sign-based and amplitude-based bit derivation methods at the same time.

**Attacks on Equiprobable Quantization** As introduced in Section 2.3.3 – similarly to the TMHD scheme – the EPQ scheme divides the distribution of counter differences $\delta_i$ into intervals of equal probability. The quantization method has to be considered public. An attacker measuring $\delta_i$ including the sign can compute the quantization, and the symbol derived on the device. Even if the attacker

**Table 5.1** Comparison of bit derivation methods regarding the secret to be protected.

| | Secret | |
|---|---|---|
| | Sign | Amplitude |
| Sign-Based [20] | x | |
| Two-Metric Helper Data [40] | | x |
| Equiprobable Quantization [42] | x | x |
| Order Encoding [29] | x | x |

can only reveal $|\delta_i|$ without the sign, only the two intervals for $|\delta_i|$ and for $-|\delta_i|$ are possible. In other words, even if protection of the sign is achieved by the temporal masking scheme, the remaining entropy for an attacker would be reduced to one bit rendering the approach useless compared to sign-based bit derivation. Therefore, EPQ is only useful if an attacker cannot learn about the amplitude of the counter difference $|\delta_i|$. Note that the argument also holds for any other ring-based PUF, for which an attacker observes the analog properties. Furthermore, it applies to other quantization schemes like equidistant quantization as well.

**Attacks on Order encoding** Lehmer-Gray order encoding from Section 2.3.3 sorts the analog values of the PUF with respect to their value followed by an encoding of the order. Using a Loop PUF with Hadamard challenges, bias is largely removed , and order encoding could be directly applied to the counter differences $\delta_i$. However, the same problem as for the other amplitude-based methods appears: An attacker observing all possible $\delta_i$ or at least $|\delta_i|$ can make strong statements about the resulting order. Therefore, not only the sign but also the amplitude of $\delta_i$ has to be protected.[1]

**Protection Methods for the Loop PUF** Table 5.1 summarizes for the different bit derivation methods the secret information of the Loop PUF, which has to be protected from SCA. For the sign-based method only the sign of the counter difference is secret. *Temporal masking* provides a low-cost countermeasure that efficiently hides the sign of the counter difference from an attacker. However, if temporal masking is applied to amplitude-based schemes, the attacker is still able to significantly reduce the entropy or to completely break the system, if $|\delta_i|$ or $\delta_i$ can be measured.

In order to protect the Loop PUF independently of the bit derivation method, either the order of challenges $C_1, \ldots, C_{N-1}$ can be randomized, i.e., an attacker does not know the correct order of her observations, or leakage about $|\delta_i|$ must be hidden in the first place. The first approach is introduced in Section 5.3 in form of *challenge randomization* that randomizes the challenge index: While the device can resolve the correct order, an attacker is not able to sort SCA observations accordingly. In Section 5.5 the second approach is taken by proposing *challenge interleaving*, a modification of the Loop PUF with low complexity. It hinders side-channel leakage without the need for additional randomness and is applicable to amplitude- and sign-based bit derivation.

## 5.2 Analysis of the Two-Metric Helper Data Scheme

This section reveals a side-channel vulnerability of the original TMHD scheme given an attacker without helper data access. A modification of the TMHD from Section 2.3.2 improves the scheme regarding mitigation of the vulnerability. Subsequently, it is shown that even with the improvements 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 5.3.

[1] Note that the same problem is expected to appear if an attacker mounts a side-channel attack on the PUFKY architecture [29] from Section 2.2.1 and is able to resolve individual RO frequencies.

### 5.2.1 Attack Vector of the Two-Metric Helper Data Scheme

In the following, the counter differences $\delta_i$ processed by the TMHD stem from a Loop PUF that is protected by the *temporal masking* scheme from Section 4.3.1. Consequently, the underlying PUF primitive is protected against SCA and only the properties of the TMHD can be targeted. First, note that for the TMHD scheme the bit value is not given by the sign of $\delta_i$, but by its amplitude. Revisiting Fig. 2.8a and Eqs. (2.4) and (2.5), which establish the bit value according to the metrics and the counter difference, if $|\delta_i| > a$ the bit value is always $r = 0$; otherwise, the bit value is $r = 1$. Thus, an attacker observing the frequencies and their difference in the side-channel can derive the bit value even in presence of the temporal masking countermeasure. This is only possible because the TMHD uses the amplitude instead of the sign to derive the PUF bits.

### 5.2.2 Formalization of the Attack Success

In the following, theoretical insights are provided into the success probability that an attacker can achieve. Note that in the formal model of the side-channel observations instead of the discretized counter difference $\delta_C$ the analog frequency difference $df_C$ is considered as it is the basis for the observed frequency difference $df_C^\star$, which is also an analog quantity.

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 $\mathcal{N}_{osc.} \sim \mathcal{N}(\mu_C, \sigma_{osc.})$ of the oscillation frequency $f_C$ that occurs from measurement to measurement. Assuming that the noise terms are normally distributed and additive, the overall noise is expressed as

$$\mathcal{N}_{attack} = \mathcal{N}_{meas.} + \mathcal{N}_{osc.} \sim \mathcal{N}(\mu_{adv.}, \sigma_{adv.}), \tag{5.1}$$

where $\sigma_{adv.} = \sqrt{\sigma_{meas.}^2 + \sigma_{osc.}^2}$.[2] Assuming that environmental conditions change slowly, any perturbation is constant among the different frequencies. Thus, systematic offsets $\epsilon$ from the nominal oscillation frequencies, i.e., $\mu_C + \epsilon$, $\mu_{\neg C} + \epsilon$, cancel out when calculating the frequency difference $df = f_C - f_{\neg C}$, and it follows that $\mu_{adv.} = \mu_C - \mu_{\neg C}$.

The following notation is adopted for the PDF of a normally distributed variable with mean $\mu$ and standard deviation $\sigma$

$$\phi^\star(x; \mu, \sigma) := \frac{1}{\sigma}\phi\left(\frac{x-\mu}{\sigma}\right) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \tag{5.2}$$

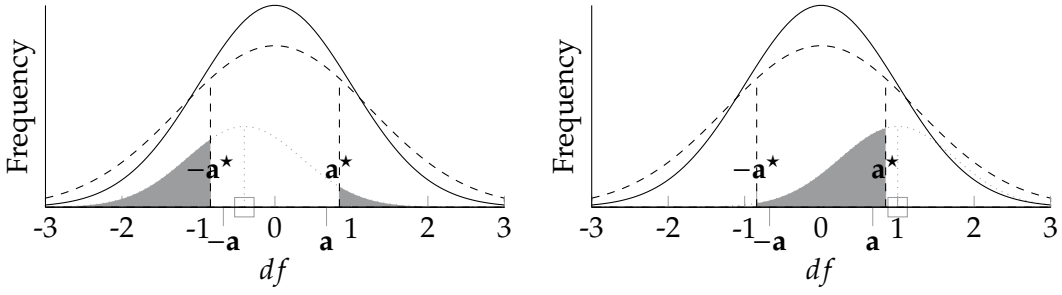
where $\phi(x)$ is the standard normal distribution.

The attacker mimics the reconstruction process by estimating bounds $\pm T1^\star$, $\pm T2^\star$ and $\pm a^\star$ from the observed values $df_C^\star$ and guesses $\hat{r}_C$ using Eq. (2.6). 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. The expected attack success decreases if the attacker observes $\sigma_{meas.} > 0$. In the following, the noise term is set to $\sigma_{osc.} = 0$ 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 $a' = a$, while the attacker does so based on the noisy versions $T1^\star$, $T2^\star$ and $a^\star$. The success probability

$$Pr_{success}(df_C, \sigma_{adv.}) = Pr[\hat{r}_C = r_C | df_C, \sigma_{adv.}] \tag{5.3}$$

defines whether an attacker can retrieve the correct PUF bit for a challenge $C$. The assumption is that the device uses $df_C$ for reconstruction, while the attacker observes $df_C^\star$ drawn from the normal distribution $\phi^\star(df^\star; df_C, \sigma_{adv.})$. In addition to the relationship of attack success and SNR, Eq. (5.3) also provides insights whether certain values $df_C$ can be more easily attacked.

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




**(a)** $|df_C| < a$ **(b)** $|df_C| \geq a$

**Figure 5.1** Visualization of the attack failure for attacker without helper data.

Weighting the success probability by the occurrence of $df$ yields the average success probability. From Section 2.3.2 $df$ is assumed to follow a normal distribution that for the sake of simplicity is transformed into a standard normal distribution for $\sigma_{osc.} = 0$. Consequently, the average success probability is given as

$$\overline{Pr}_{success}(\sigma_{adv.}) = \int_{-\infty}^{\infty} Pr_{success}(df, \sigma_{adv.}) \cdot \phi^{\star}(df; 0, 1)\, \mathrm{d}df. \tag{5.4}$$

### 5.2.3 Exploiting the Two-Metric Helper Data Scheme

In the following, side-channel vulnerabilities of the TMHD from Section 2.3.2 with and without helper data knowledge are investigated.

#### Side-Channel Analysis Without Helper Data Access

In a first step it is shown that even if an attacker does not know the helper data, i.e., whether metric $M1$ or $M2$ is applied, the TMHD leaks side-channel information. From Fig. 2.8a the attacker needs to observe the amplitude of the frequency difference $df^{\star}$ regarding $a^{\star}$: if $|df^{\star}| > a^{\star}$, the guessed PUF bit is $\hat{r}_C = 0$, otherwise $\hat{r}_C = 1$. The success probability that the estimated PUF bit $\hat{r}_C$ matches the correct $r_C$ is

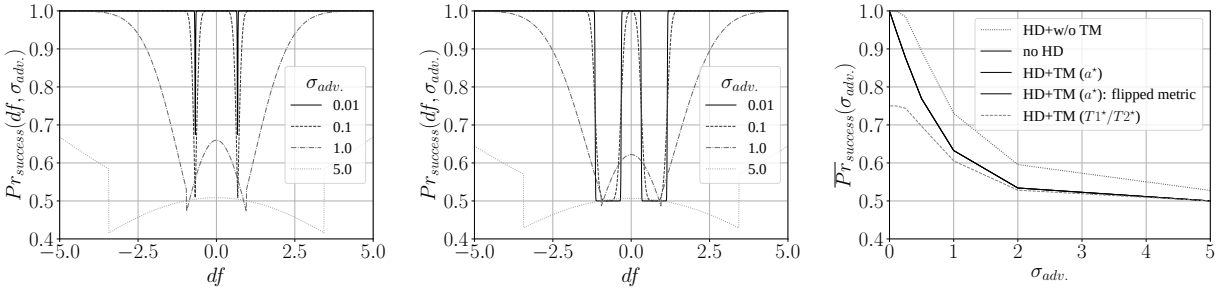
$$\begin{aligned} Pr^{no\,HD}_{success}(df, \sigma_{adv.}) &= 1 - Pr[\hat{r}_C \neq r_C] \\ &= 1 - \left(Pr\left[|df^{\star}| \leq a^{\star} \,\middle|\, |df| > a\right] + Pr\left[|df^{\star}| > a^{\star} \,\middle|\, |df| \leq a\right]\right) \\ &= 1 - \begin{cases} \int_{-\infty}^{-a^{\star}} \phi^{\star}(df^{\star}; df, \sigma_{adv.})\, \mathrm{d}df^{\star} + \int_{a^{\star}}^{\infty} \phi^{\star}(df^{\star}; df, \sigma_{adv.})\, \mathrm{d}df^{\star}, & -a \leq df < a \\ \int_{-a^{\star}}^{a^{\star}} \phi^{\star}(df^{\star}; df, \sigma_{adv.})\, \mathrm{d}df^{\star}, & -a > df \geq a \end{cases}. \end{aligned} \tag{5.5}$$

Figs. 5.1a and 5.1b depicts the distributions of $df/df^{\star}$ from which device and attacker derive their bounds $a/a^{\star}$ as solid and dashed curves respectively. Due to the additional noise term the attacker faces, the dashed distribution is notably broader. The dotted curve represents the distribution of observed values $df_C^{\star}$ for an enrolled value $df_C$ marked as square. The filled area below the dotted curve marks a failed attack according to Eq. (5.5).

Fig. 5.2a depicts the success probability depending on the enrolled value of $df_C$ for different levels of noise. For $df_C^{\star} \approx |a^{\star}|$, the attacker faces the biggest uncertainty as the additional noise $\sigma_{adv.}$ changes the retrieved PUF bit most easily close to the decision boundary. Note that the attack does not change whether the *temporal masking* countermeasure from Section 4.3.1 is applied or not – in both cases, the amplitude reveals the PUF bit.

#### Side-Channel Analysis With Helper Data Access

Finally, SCA with helper data access is considered, i.e., the attacker can combine observations $df_C^{\star}$ with the corresponding bit derivation helper data $w_C^{bd}$. In case no temporal masking is applied, the

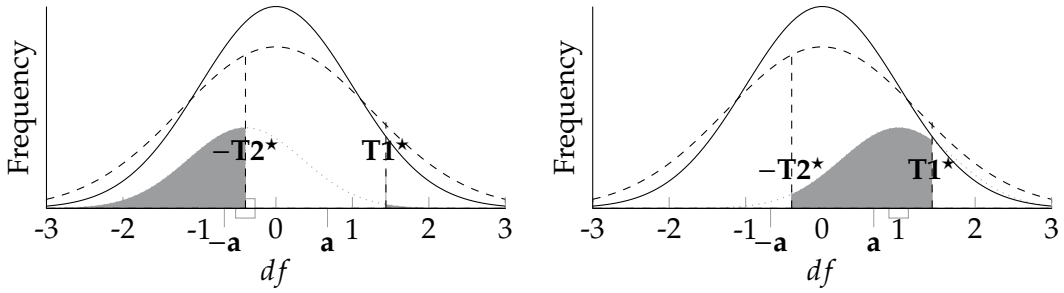



**(a)** no helper data / temporal masking (using $a^\star$)

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

**(c)** Average success rate

**Figure 5.2** Simulated attack success probability for different levels of attacker noise. (a) – (b) Depending on the enrolled value $df$. (c) Integrated success probability according to the occurrence of $df$.




**(a)** $-a < df_C < 0$

**(b)** $df_C \geq a$

**Figure 5.3** 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 $M1$.

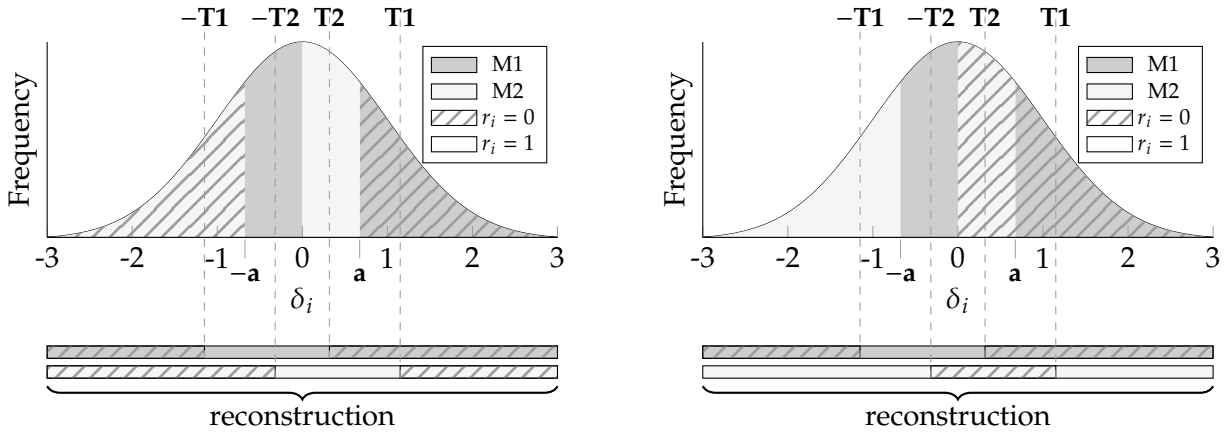
additional information can be leveraged to improve the attack as outlined in Appendix A. This highlights that the TMHD scheme without any protection is prone to SCA attacks and the bit derivation helper data $\mathbf{w^{bd}}$ improves the reliability of the PUF as well as the attack success. As without temporal masking the frequency difference $df_C$ would be revealed independently of the helper data scheme, in the following temporal masking is considered.

Note that if temporal masking is activated, the attacker cannot trust the sign of the observed frequency difference $df_C^\star$. 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. The effect is neglected 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 $|df^\star|$, and the attack success rate is equal to the case in which no helper data is known. Second, the attacker can try to exploit the helper data by using the bounds $T1^\star$ and $T2^\star$ estimated from the attack. However, it has to be considered that due to the temporal masking countermeasure the sign of the observation, which is compared the bounds, could be flipped. Taking metric $M1$ as an example, the attacker would average the choice of $-T1^\star$ and $T2^\star$ (as defined by $M1$ in Fig. 2.8a) and $-T2^\star$ and $T1^\star$ (reflecting a sign flip depicted as in Fig. 5.3) and the probability for an attack error is

$$\begin{aligned} P_1(df, \sigma_{adv.}) &= Pr[\hat{r}_C \neq r_C | w_C^{bd} = M1, df > a] \\ &= \frac{1}{2}\left[\int_{-T1^\star}^{T2^\star} \phi^\star\left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star + \int_{-T2^\star}^{T1^\star} \phi^\star\left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star\right]. \end{aligned} \tag{5.6}$$

The other intervals from the enrollment are covered accordingly. Fig. 5.3 depicts the resulting error if the observed $df^\star$ has flipped sign. Fig. 5.2b highlights that the intervals between $-T1^\star$ and $-T2^\star$,




**(a)** Original TMHD **(b)** Modified TMHD

**Figure 5.4** TMHD bit derivation from counter differences $\delta_i$ with different choices of metrics. (a) Original metrics [40] as in Fig. 2.8b, (b) flipped metric $M2$.

and $T1^\star$ and $T2^\star$ are most prone to errors. Note that for increasing attacker noise $\sigma_{adv.}$ the intervals $[-T1^\star, T2^\star]$ and $[-T2^\star, T1^\star]$ overlap increasingly.

In Fig. 5.2c the overall success rate for varying noise levels $\sigma_{adv.}$ is depicted according to Eq. (5.4) for the different scenarios outlined in this section. 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 $\pm a^\star$. Using the bounds $\pm a^\star$, the average error in Fig. 5.2c is equal to the attack success for the case without helper data and no temporal masking, i.e., no additional information is achieved from the helper data if temporal masking is applied. From Fig. 5.2c, if temporal masking is disabled, helper data access improves the attack even further.

Summing up, the TMHD is prone to SCA and temporal masking does not hinder these attacks. Therefore, either the TMHD scheme requires a modification or a dedicated countermeasure.

### 5.2.4 Hardening of the Two-Metric Helper Data Scheme

A first attempt to mitigate SCA of the TMHD is to modify the mapping of the metrics. If the bit value with metric M2 is inverted compared to Eq. (2.4), i.e.,

$$M1 : r_i = \begin{cases} 0, & T2 \leq \delta_i \vee \delta_i < -T1 \\ 1, & -T1 \leq \delta_i < T2 \end{cases} \qquad M2 : r_i = \begin{cases} 1, & T1 \leq \delta_i \vee \delta_i < -T2 \\ 0, & -T2 \leq \delta_i < T1 \end{cases}, \tag{5.7}$$

the PUF response is related to the sign of the counter difference $\delta_i$ as shown in Fig. 5.4b. Note that the choice of the metric, i.e., the helper data, is maintained according to Eq. (2.5) as the comparison to the original TMHD scheme in Fig. 5.4a shows.

As the amplitude no longer reveals information, temporal masking protects the TMHD 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 $df_C^\star$ is randomly altered, consider the absolute values $|df^\star|$ and estimate the parameters $\pm a^\star$, $\pm T1^\star$ and $\pm T2^\star$ as described in Section 5.2.2. 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

$$\hat{r}_C = \begin{cases} 0, & \left(|df_C^\star| > a^\star \wedge w_C^{bd} = M1\right) \vee \left(|df_C^\star| \leq a^\star \wedge w_C^{bd} = M2\right) \\ 1, & \left(|df_C^\star| > a^\star \wedge w_C^{bd} = M2\right) \vee \left(|df_C^\star| \leq a^\star \wedge w_C^{bd} = M1\right) \end{cases}. \tag{5.8}$$

Using Eq. (5.8), the attacker achieves the same success probability as for the non-flipped metric as depicted in Fig. 5.2c. Nevertheless, using the sign instead of the amplitude to derive PUF bits, the following improvements are achieved:

**Table 5.2** Remaining entropy in bits after smart guessing on the 63-bit Loop PUF using TMHD 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 |

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 similar to Section 4.1.4 can be derived: In Eq. (5.8), the closer $|df_C^\star|$ is to the boundary $a^\star$, where 0 and 1 change, the lower is the reliability of the estimate. The next section provides practical results from side-channel measurements to verify the possibility of a successful SCA attack.

### 5.2.5 Practical Results for SCA of the Two-Metric Helper Data Scheme

The same setup with a 63-bit Loop PUF and the approach from Section 4.1 are used to retrieve estimates of the frequency differences $df_C^\star$. The attack from Section 5.2.4 is carried out by first performing an enrollment on the actual counter values averaged over ten runs. From the enrollment, a set of reference bit derivation helper data $\mathbf{w^{bd}}$ is generated. The measurements of challenge pairs $(C, \neg C)$ are randomized to emulate the temporal masking countermeasure.

For the following practical results, the TMHD bounds used for the attack are derived on the measured frequencies, i.e., the attacker faces additional noise compared to the device according to Eq. (5.1). Table 5.2 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 first alters bits derived from frequency differences $|df_C^\star|$ close to $a^\star$ and is stopped at 20 bits 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 TMHD even with the flipped metric from Section 5.2.4 with reasonable effort.

In conclusion, a modification by flipped metrics does not hamper SCA of the TMHD and the following sections investigate possible countermeasures.

## 5.3 Protection of the Two-Metric Helper Data Scheme by Challenge Randomization

According to Section 5.2 an attacker with helper data knowledge and side-channel observations of frequencies can break the TMHD scheme. The prerequisite is that for each observed frequency difference $df_i^\star$ the corresponding bit derivation helper data $w_i^{bd}$ is known. Consequently, hindering the mapping of observations and helper data by randomizing the measurement order or, equivalently the order of challenge pairs, provides protection against SCA. As in a PUF scenario with publicly stored helper data protected memory contradicts the use case, storing a randomized mapping of helper data and challenges securely is not possible. Further, by Kerckhoffs's principle, the attacker knows how the challenges are generated and applied. Therefore, this section investigates *Challenge Randomization* countermeasures to randomly permute challenge pairs during reconstruction. The countermeasures are derived as general as possible, but are exemplified for a 63-bit Loop PUF in accordance with Sections 4.1 and 5.2.5.

### 5.3.1 Attack Vector of the Protection Mechanism

In the following, the flipped metric as introduced in Section 5.2.4 is used and temporal masking is enabled. Thus, the attacker measures frequency differences $df_i^\star$ of the Loop PUF without knowing the sign. Even if the order of the different $df_i^\star$ is randomized, she still knows *which* frequency differences are used. The frequency differences can be distinguished by their absolute values and further by the pairs of observed frequencies $\hat{f}_{C_i}$ and $\hat{f}_{\neg C_i}$. Thus, a divide and conquer strategy is possible:

1. bring the frequency differences into the order of the helper data
2. mount the attack from Section 5.2.5 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 5.2.5 is performed. The remaining entropy after a single measurement is determined by noise. As a consequence, the complexity to bring the frequency differences into the correct order mainly determines the difficulty of the attack.

According to Section 2.2.2, a Loop PUF with $N$ delay stage generates $N-1$ secret bits from challenge pairs $(C_i, \neg C_i)$ and their corresponding counter differences $\delta_i$. There exist $N-1!$ different permutations of the order of challenge pairs, and for $N > 4$ permutation suffices to protect the secret as $N-1! > 2^{N-1}$. The difficulty is to develop an algorithm, which generates the permutations or a subset of permutations efficiently and unpredictably.

### 5.3.2 True Random Number Generator-Based Protection

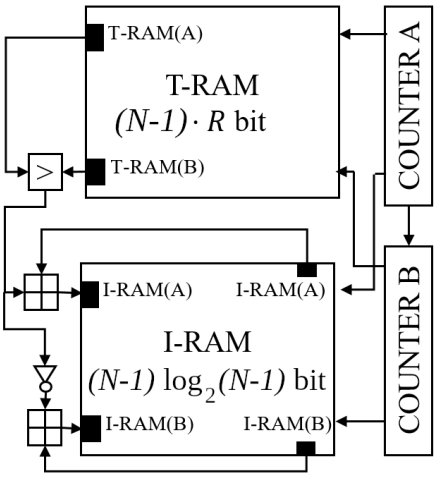
The first approach is a countermeasure that randomizes the order of challenge pairs $(C_i, \neg C_i)$ using a TRNG. For this purpose at least $N-1$ random numbers of $R \geq \lceil \log_2(N-1) \rceil$ bits are needed to generate the index $i \in \{0, \ldots, N-2\}$ that selects the corresponding challenge pair. In this case, an attacker with SCA knowledge observes a random permutation of frequency differences, which hinders the mapping of helper data and observations, and eventually impedes an attack. Two questions are addressed in the following:

(i) How to map from random values to a unique sequence of challenge pairs?

(ii) What is a suitable choice of $R$ to mitigate SCA attacks while retaining a low implementation overhead?

#### Mapping of Random Values to a Unique Sequence

An efficient method to solve the mapping of random values to a unique sequence is given in Fig. 5.5. A possible implementation, depicted as block diagram in Fig. 5.5a, requires two $\lceil \log_2(N-1) \rceil$-bit adders, two $\lceil \log_2(N-1) \rceil$-bit counters, an $R$-bit comparator and $(N-1) \cdot \left( \lceil \log_2(N-1) \rceil + R \right)$ bit of storage capacity. The permutation generator, which is related to Lehmer encoding, takes as an input $N-1$ 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 $\frac{(N-1)\cdot(N-2)}{2}$ 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 $0, \ldots, N-2$. The sorting algorithm provided in Fig. 5.5b defines – independent of the number of random bits $R$ – the permuted order of all challenge pairs. The order is unpredictable for the attacker, but can be resolved by the device in order to match counter differences and helper data.




**(a)** Block diagram

**Algorithm 1: Permutation Generator**

Choose $N-1$ distinct $R$-bit random numbers $x_i$
Store $x_i$ to `T-RAM`.
**for** `A` in 0 to $N-2$ **do**
  **for** `B` in `A`+1 to $N-1$ **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`.

**(b)** Algorithm

**Figure 5.5** TRNG-based permutation generator.

**Table 5.3** Probability for a collision of random numbers for $N-1 = 63$ rounded to three digits.

| $R$ | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | ... | 20 | 21 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p_{col}$ | 1.000 | 1.000 | 0.999 | 0.975 | 0.842 | 0.603 | 0.370 | 0.206 | 0.109 | ... | 0.002 | 0.001 |

#### Randomness Required by the TRNG-Based Protection

If a TRNG provides only distinct random numbers, the protection mechanism does not reveal information about the sorting. In this case, $(N-1)\cdot R$ random bits would be required, e.g., $63\times 6 = 378$ random bits for a 63-bit Loop PUF. However, this is the lower limit as in practice collisions are possible, i.e., the same random number may be sampled twice. Options to overcome the problem of repeated random numbers include to put the challenges for which collisions appear into a predefined order or to re-sample in case of a collision. A predefined order of challenges leads to a higher probability for specific permutations that gives additional information to an attacker. Therefore, the re-sampling approach is followed as it prevents attacks that exploit permutations with distinct probabilities. Re-sampling generates a new random number if a collision appears, and the amount of required additional random numbers is analyzed in the following.

Similar to the birthday problem, the probability that from the set of $2^R$ possible random numbers at least two collide when generating $N-1$ numbers is

$$p_{col} = 1 - \left(1 - \frac{1}{2^R}\right)^{(N-1)(N-2)/2} . \tag{5.9}$$

Table 5.3 summarizes the corresponding probabilities for $N-1 = 63$, where only for random numbers of $R \geq 21$ bits the probability for a collision of two number is less than one in a thousand. In other words, decreasing the probability of a collision is costly in terms of the required number of random bits.

However, it is *less important if collisions* appear *than how many bits* are required when generating a set of distinct random numbers if collisions are resolved through re-sampling. Assume that the random numbers $x_i$ are sampled sequentially and that the current $x_i$ is re-sampled until the TRNG provides a number not yet used. Further, assume an ideal TRNG providing $R$-bit outputs such that all $2^R$ possible sequences are equally likely. Then, the probability for a collision for the $i^{\text{th}}$ random number with $0 \leq i \leq N-2 < 2^R$ is

$$p_{re,R}(i) = \frac{i}{2^R}, \tag{5.10}$$

**Table 5.4** TRNG-based countermeasure: Optimum number and bit lengths of random numbers and required re-sampling.

| $N-1$ | 15 | 31 | 63 | 127 | 255 | 511 | 1023 |
|---|---|---|---|---|---|---|---|
| $E_{bits,N-1}$ (bit) | 99.03 | 247.36 | 577.23 | 1312.17 | 2930.88 | 6464.41 | 14122.15 |
| $R$ (bit) | 5 | 7 | 8 | 9 | 10 | 11 | 12 |
| $E_{re,R}$ | 4.81 | 4.34 | 9.15 | 18.80 | 38.09 | 76.67 | 153.85 |

i.e., the more random numbers are drawn, the higher is the probability for a collision with previous random numbers. The expected number of samples required to receive a random number $x_i$ that has not been used is

$$E_R(i) = \frac{1}{1 - p_{re,R}(i)} = \frac{1}{1 - \frac{i}{2^R}} = \frac{2^R}{2^R - i}. \tag{5.11}$$

Summing Eq. (5.11) for all successively drawn random bits yields the overall expected number of samples. As a consequence, the average number of required random bits when sampling $N-1$ random numbers of length $R$ is

$$E_{bits,N-1}(R) = R \cdot \sum_{i=0}^{N-2} E_R(i), \tag{5.12}$$

which has a minimum in $R$. For a given $N-1$ the minimum defines the optimal choice of $R$ regarding the expected amount of TRNG bits needed. However, this comes at cost of the average number of re-samplings, i.e., the number of expected samples above the minimum possible number of samples

$$E_{re,R} = \left( \sum_{i=0}^{N-2} E_R(i) \right) - (N-1). \tag{5.13}$$

Table 5.4 provides the optimum average number of random bits and the corresponding bit length $R$ for different $N-1$, which can be used for the permutation generator from the previous section. Furthermore, the required average re-samplings are given. Considering the example of the $N-1 = 63$-bit Loop PUF the minimum is reached at $R = 8$, i.e., on average $E_{bits,63}(8) \approx 577.23$ bits are needed, which is substantially more than the minimum of 378 bits without re-sampling. Note that the number of re-samplings and consequently the runtime of the generation is not constant, but can only be provided on average. On the one hand, if the generation of an $R$-bit random number takes less time than the Loop PUF needs to derive a bit, re-sampling could be done in parallel to the bit derivation. On the other hand, there is a non-zero probability that re-sampling has to be repeated more than once per random number and takes longer than the bit derivation, i.e., a constant time behavior can not be guaranteed.

Summing up, the drawback of the TRNG-based permutation generator is the significant amount of random bits. Furthermore, due to re-sampling of random numbers, the runtime is non-deterministic.

### 5.3.3 Towards a Lightweight Protection of the Two-Metric Helper Data Scheme

The TRNG-based approach from Section 5.3.2 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 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, and lower implementation complexity is feasible that retains a sufficient level of protection.

This section introduces a lightweight implementation based on LFSRs for Loop PUFs with the number of stages chosen at powers of two. In this case, $2^q - 1$ challenge pairs have to be permuted,

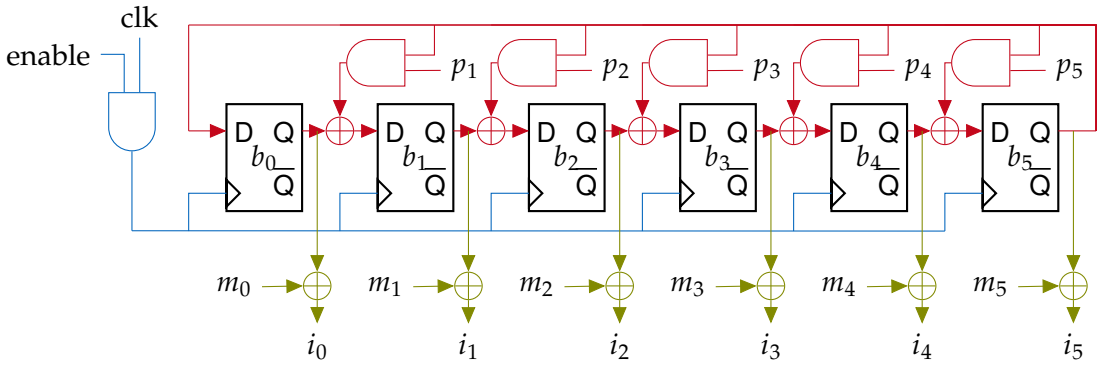



**Figure 5.6** Combined low-complexity countermeasures to generate 6-bit indices for a 63-bit Loop PUF.

and the $2^L - 1$ states of an $L = a$-bit LFSR directly represent the index of a challenge. In Fig. 5.6 an LFSR with six state bits $b_0$ to $b_5$ is depicted. The approach uses a combination of the LFSR seed ($b_0$ to $b_5$), clock shifting (in blue), an additive mask (in green), and the randomized LFSR feedback polynomial (in red). The single parts and their expected entropy are investigated in the following.

**Entropy From the Random Seed** The first randomization technique is the use of a random initialization of the LFSR state bits. Since all states of the LFSR are used to index the $N - 1$ 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 $\log_2(N - 1)$ bit of entropy, i.e., 5.98 bits in case of the 6-bit LFSR used by the 63-bit Loop 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 $n^{\text{th}}$ shift, where $n$ is selected at random once per reconstruction. In Fig. 5.6 the enable signal is set for $n$ cycles after which the values of the challenge bits $i_0, \ldots, i_5$ are used to determine the challenge index. As the LFSR output is cyclic, the shift by $n$ has to be relative prime to the period of the LFSR in order to reach all states of the LFSR. Euler's totient function $\varphi(N - 1)$ provides the number of relative primes to $N - 1$. In case of the 6-bit LFSR there are 36 values for $n$ corresponding to approximately 5.17 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 to observe the LFSR is small: 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. The attacker can observe the clocking only $N - 1$ 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 modify the state of the LFSR by applying an additive mask corresponding to $\log_2 N$ bits of entropy. By adding a mask the index zero is introduced if the mask value is the same as the LFSR state value. However, the index equal to the mask disappears as the LFSR state cannot be zero. As the all-zero challenge $C_0$ is not used for response derivation in the Loop PUF, the zero index is mapped to the missing index in the analysis.

**Entropy From the Random Polynomials** As a fourth method the feedback polynomial is randomly chosen for the LFSR. In order to assure sequences of maximum length that cover all indices, primitive

**Table 5.5** Entropy and required randomness of combined countermeasure compared to TRNG-based approach.

| | | | $N-1$ | | | | |
|---|---|---|---|---|---|---|---|
| Mechanism | Effect | Possibilities | 15 | 31 | 63 | 127 | 255 |
| LFSR Seed | Offset | $N-1$ | 3.91 bits | 4.95 bits | **5.98 bits** | 6.99 bits | 7.99 bits |
| Multi-Bit Shifting | Shifting | $\varphi(N-1)$ | 3 bits | 4.91 bits | **5.17 bits** | 6.98 bits | 7 bits |
| Masking | XOR | $N$ | 4 bits | 5 bits | **6 bits** | 7 bits | 8 bits |
| LFSR Polynomial | Re-ordering | $\frac{\varphi(N-1)}{\log_2 N}$ | 1 bit | 2.58 bits | **2.59 bits** | 4.17 bits bit | 4 bits |
| Summed Entropy | | | 11.91 bits | 17.45 bits | **19.73 bits** | 25.14 bits | 26.99 bits |
| Required Randomness | | | 12 bits | 18 bits | **21 bits** | 26 bits | 27 bits |
| TRNG: Entropy | | | 15 bits | 31 bits | **63 bits** | 127 bits | 255 bits |
| TRNG: Av. Randomness | | $\geq (N-1)\lceil\log_2(N-1)\rceil$ bits | 99.03 bits | 247.36 bits | **577.23 bits** | 1312.17 bits | 2930.88 bits |

polynomials of degree $\log_2 N$ over the finite field $GF(2)$ have to be used. The number of different primitive polynomials is $\varphi(N-1)/\log_2 N$,[3] i.e., for the 63-bit Loop PUF, there are six primitive polynomials of degree $\log_2 64 = 6$ over the finite field that lead to approximately 2.59 bits of entropy.

### 5.3.4 Summary of Countermeasures

The idea of the combined countermeasure is to increase the complexity for SCA while maintaining a low complexity of the countermeasure. Table 5.5 summarizes the four proposed protection mechanisms for different $N-1$-bit Loop PUFs, where the 63-bit Loop PUF is highlighted in bold. The summed entropy for all countermeasure mechanisms is given and the required randomness is provided assuming that the randomness for each countermeasure is acquired separately, i.e., rounding to the next integer per countermeasure. Compared to the TRNG-based countermeasure from Section 5.3.2, the number of required bits is significantly reduced at the cost of a reduction of the entropy, i.e., less than $N-1$ bits are achieved. However, recall that the attacker has to perform the smart guessing attack from Section 5.2.5 for the provided possibilities if the different sequences are indistinguishable. For the 63-bit Loop PUF, summing the entropy values of the four protection mechanisms yields 19.73 bits of entropy from a 6-bit LFSR. Instead of 577.23 bits on average for the TRNG-based countermeasure, the combined lightweight countermeasures require only 21 bits of randomness. While the brute-force complexity below 20 bits would not prevent an attack, 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.

The practical security analysis in Section 5.4 investigates how an adversary can distinguish sequences generated by the lightweight countermeasure in order to reduce the entropy and enable an attack.

## 5.4 Security Analysis of the Lightweight Challenge Randomization Countermeasure

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 5.2 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.

[3] `https://archive.ph/UA0Iq`, last accessed 28th September 2022.

This section analyzes the quality of the lightweight countermeasure from Section 5.3.3 in order to show its limitations and to point towards possible solutions. The different countermeasures are discussed individually and followed by practical evaluation results of individual and combined countermeasures. In the following, a 6-bit LFSR is considered, and the Galois LFSR state from Fig. 5.6 is interpreted as an integer, e.g., $[1, 0, 0, 0, 0, 0]$ corresponds to 1.

### 5.4.1 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 $df_i^\star$ of the PUF as symbols $s_i,\ i \in 1, \ldots, N-1$. She knows that there is a native order $\mathbf{s}_{nat} = [s_1, \ldots, s_{N-1}]$ of the symbols, which matches the sorting of the bit derivation helper data $\mathbf{w^{bd}} = \left[w_1^{bd}, \ldots, w_{N-1}^{bd}\right]$. 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 of the combined countermeasures in Section 5.3.3, more than 850000 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 $\mathbf{s}_{nat}$. For noisy measurements, a direct matching of the resorted sequences is not possible as the observed frequencies vary across measurements. However, the attack can be modified by correlating the two observed and transformed sequences $\mathbf{s}_{obs,1}\mathbf{A}_1^{\star -1}$ and $\mathbf{s}_{obs,2}\mathbf{A}_2^{\star -1}$, i.e., by using the similarity of two corrected sequences.

The following analysis determines how many solutions $\mathbf{s}_{cand}$ exist, i.e., how much entropy the attacker faces.

### 5.4.2 Theoretical Analysis

Each of the different lightweight countermeasures causes a permutation of the native order $\mathbf{s}_{nat}$. The effect of each countermeasures is summed up in Table 5.5. The following theoretical analysis provides insights how far an attacker can deduce the native order from two permuted sequences determined from observations. The analysis considers the best case of noise-free observations, i.e., the attacker does not face any uncertainties about the observed symbols.

**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 $m$ bits 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 $\alpha$ and $\beta$ from their seeds, the relative shift of the sequences corresponds to a $\kappa$-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 $\mathbf{T}_\alpha$ correspond to the permutation of the LFSR state sequence under shifts by $\alpha$, and let $\mathbf{T}_\beta$ correspond to the permutation of the LFSR state sequence under shifts by $\beta$. 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 bit-wise XOR onto the LFSR state with the all-zero result mapped to the mask value. Different from the previous methods, two permutations $M_\alpha$ and $M_\beta$ 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 uncertainty for 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 previous countermeasure, polynomials do not lead to an increased uncertainty for the attacker but rather allow for a better attackability of the LFSR. The reason is, that the permutation matrices $P_\alpha$ and $P_\beta$ 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.

### 5.4.3 Practical Evaluation

The theoretical insights from Section 5.4.2 are verified with experimental data from synthetic symbols as well as on the measurement data of campaign #1 used in Table 5.2. The synthetic symbols and measured frequency differences are permuted in software with different permutation strategies from Section 5.3.3 enabled. 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. In accordance with the attack strategy, all inversions are pre-computed to map from a permutation back to the native sorting. For the 6-bit LFSR, a list with 870912 inversions is generated. Then, the two observed sequences of symbols are permuted with respect to to the pre-computed inversion list and are correlated with each other. A correlation of $\rho = 1$ would be a perfect match of sequences, which only occurs for synthetic data. As frequency differences change slightly from measurement to measurement, for experimental data the correlation $\rho < 1$ depends on the noise level. Note that the complete attack takes on a commodity computer[4] 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 5.6 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**). Results are shown for noise-free simulated data with $\rho = 1$ and for the lowest ($\rho_{min} = 0.91$) and highest

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

**Table 5.6** Results for SCA of the 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 | | | | Simulated Data (Noise-free) | | | Measurement Data (Noisy) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | $\rho_{max} = 0.97$ | | | $\rho_{min} = 0.91$ | | |
| **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 | - | - | 2268 | 2268 | 2268 | 2268 | 2268 | 2268 | 2268 | 2268 | 2268 |
| x | x | x | - | 378 | 378 | 756 | 378 | 378 | 3780 | 378 | 2268 | 10584 |
| x | x | - | x | 2 | 2 | 13608 | 2 | 4 | 13608 | 2 | 4 | 13608 |
| x | x | x | x | 2 | 4 | 2268 | 2 | 4 | 11340 | 4 | 13 | 870912 |

($\rho_{max} = 0.97$) noise observed in the data set. 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 $\rho' \geq \rho - \varepsilon$ with the data. The correlation $\rho$ considers the noise level of the data sets and setting $\varepsilon = 10^{-6}$ prevents rounding errors. The values indicate, how many times the attacker would have to run the attack on the TMHD scheme from Section 5.2.4 under different mappings between helper data and frequency differences.

Some remarks regarding the results in Table 5.6:

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 (**R**, **T**, **P**) is always $13608 = 36 \cdot 63 \cdot 6$, both corresponding to the theoretical maximum according to Section 5.3.3. 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 with the other mechanisms.

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., (**R**, **T**, **P**) vs. (**R**, **T**, **M**, **P**), and (**R**, **T**) vs. (**R**, **T**, **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 $63 \cdot 36 = 2268$ indistinguishable sequences when observing ten different pairs of Loop PUF measurements. In other words, instead of 19.73 bits from the combination of four countermeasures, only 11.15 bits of protection are achieved. While 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 2268 most likely results or drop results, which have more than 2268 equally high correlations. The resulting up to

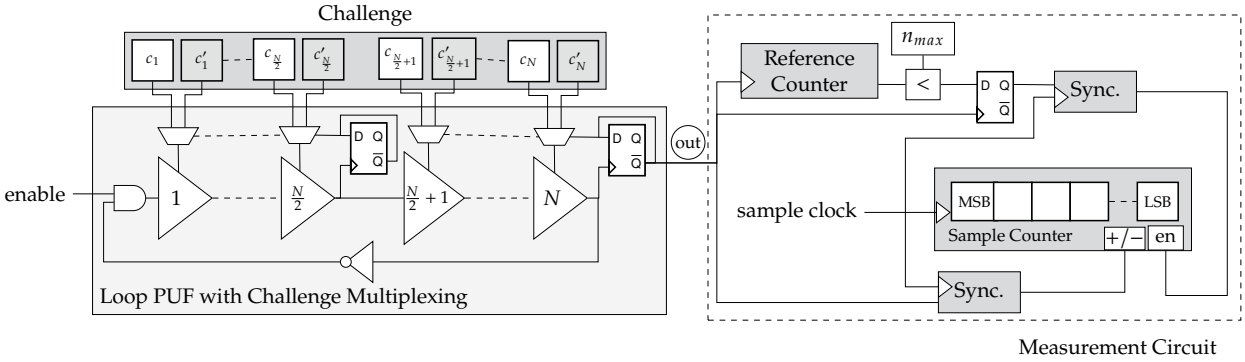



**Figure 5.7** Schematic of challenge interleaving and time domain measurements of the ICLooPUF. ©2022 IEEE [21]

$N_{meas}! \times 2268$, sequences could be used in parallel to match helper data and frequency differences and to run the attack from Section 5.2.5. This eventually demonstrates that the lightweight countermeasure hardly provides sufficient protection for the TMHD scheme. Nevertheless, the discussion highlights pitfalls, e.g., regarding combined permutations, and provides indicators on how to develop improved lightweight protection mechanisms in the future.

In order to protect the Loop PUF also for amplitude-based bit derivation, the following section introduces architectural modifications of the primitive that aim at preventing exploitable side-channel emanations in the first place.

## 5.5 The Interleaved Challenge Loop PUF: A Side-Channel Hardened PUF Primitive

This section proposes and analyzes the Interleaved Challenge Loop PUF (ICLooPUF), an enhancement of the Loop PUF. The goal of the design modifications is to harden the Loop PUF against SCA of amplitude-based bit derivation such as the attack on the TMHD scheme in Section 5.2. In Section 5.2 it has been shown that the TMHD is not protected against SCA by the *temporal masking* countermeasure from Section 4.3.1 as the absolute value of a frequency difference is used in the secret-bit derivation, and randomization of the sign provides no protection. *Challenge randomization* with a TRNG provides protection, but requires a substantial amount of random bits, does not have a guaranteed maximum runtime, and exhibits a significant complexity increase. Therefore, to provide robustness against SCA with low overhead for amplitude-based bit derivation schemes like TMHD, this section proposes the dynamic change of challenges, termed as *challenge interleaving*, for the Loop PUF, resulting in the ICLooPUF. The proposed technique thwarts SCA by breaking the relation between spectral emanations of the implementation and the frequency difference of the challenges, i.e., the secret response.

A proof-of-concept implementation on an FPGA shows the practical feasibility, followed by the theoretical validation of the resistance of the ICLooPUF against SCA. Finally, practical evidence is provided that the new primitive does indeed not reveal exploitable side-channel leakage.

### 5.5.1 Architecture of Challenge Interleaving

This section introduces the ICLooPUF, an SCA-hardened enhancement of the Loop PUF described in Section 2.2.2. The fundamental difference is that instead of applying the challenges $C$ and $\neg C$ sequentially, they are applied in an interleaved manner. This is, within a single measurement run, both challenges are alternatingly applied. When using challenge interleaving, a different measurement concept must be applied, too. The resulting ICLooPUF is shown in Fig. 5.7.

**Table 5.7** Comparison of hardware resources. ©2022 IEEE [21]

| | Loop PUF | ICLooPUF |
|---|---|---|
| Flip-flops for... | | |
| ... challenge interleaving | – | $2 - N$ |
| ... synchronization | – | 5 |
| 2-to-1 MUXes for.. | | |
| ... path selection | $N$ | $N$ |
| ... challenge interleaving | – | $N$ |
| Reference counter | 1 | 1 |
| Sample counter | 1 | 1 |

Similar to the Loop PUF, the ICLooPUF consists of an oscillator formed by configurable delay stages and an inverting feedback. When the enable signal is triggered, a rising edge propagates through the delay chain starting from the AND gate in Fig. 5.7. Any node in the delay chain passed by the rising edge is set to logical 1 until the falling edge arrives. Assume switching of the challenge bit, which defines the path through a delay element, as soon as the rising edge has passed the input and output nodes of a delay element. Under this assumption, the wave traversing through the ring passes for one period the delay elements defined by challenge $C = [c_1, \ldots, c_N]$, and for one period the delay elements defined by challenge $\neg C = [c'_1, \ldots, c'_N]$. The challenge bits are switched by T-FFs triggering on the rising edge traversing through the ring. The T-FFs control the multiplexing of the two challenge bits to be interleaved per delay stage.

There is no need to switch every delay stage individually. In particular, it is possible without side effects to switch any delay element with input and output stabilized to a fixed value, which could be achieved by only two T-FFs as in Fig. 5.7. However, since the propagation of the signal is an analog process, the voltage of several nodes can be on an intermediate voltage levels at the same time. Switching such nodes can cause glitches resulting in additional rising edges and effectively errors in the derived secret. Therefore, the number of T-FFs must not be selected too low. A suitable number of T-FFs is discussed for the proof-of-concept design in Section 5.5.3.

Summarizing, Table 5.7 depicts the hardware resources of the ICLooPUF compared to the original Loop PUF, showing that the overhead is very limited. Furthermore, the ICLooPUF adds little design costs as only few components require manual routing, which is practically verified in the implementation in Section 5.5.3.

**PUF Entropy Extraction** In the original Loop PUF construction from Section 2.2.2 a measurement counter determines the oscillations under a specific challenge $C$, and a reference counter stops the counting after a predefined time $T_{acq} = n_{acq}/f_{ref}$. According to Eq. (4.1) the value of the measurement counter $v_C$ is proportional to the Loop PUF frequency.

In the ICLooPUF, challenges $C$ and $\neg C$ are interleaved, i.e., two alternating periods of length $T_C$ and $T_{\neg C}$ are present at the output. Without challenge interleaving, the expected frequencies of the oscillation are $f_C = \frac{1}{T_C}$ and $f_{\neg C} = \frac{1}{T_{\neg C}}$ for $C$ and $\neg C$. Using the measurement concept of the original Loop PUF would result in a counter value related to the average frequency $\bar{f}_{C,\neg C}$ through

$$\bar{T}_{C,\neg C} = \frac{T_C + T_{\neg C}}{2} \Rightarrow \bar{f}_{C,\neg C} = \frac{2 \cdot f_C \cdot f_{\neg C}}{f_C + f_{\neg C}}. \tag{5.14}$$

However, the goal of the ICLooPUF is not to measure the average frequency but to directly get the difference $T_C - T_{\neg C}$ between the period lengths under challenges $C$ and $\neg C$. Therefore, different from the original approach, the difference of the two oscillations is measured in the time domain.

For this purpose, in the measurement circuit in Fig. 5.7 a sample clock with frequency $f_s \gg \max\{f_C, f_{\neg C}\}$ is applied. The sampling frequency is limited to $2 \cdot \max\{f_C, f_{\neg C}\} < f_s \leq f_{clk}$, i.e., by

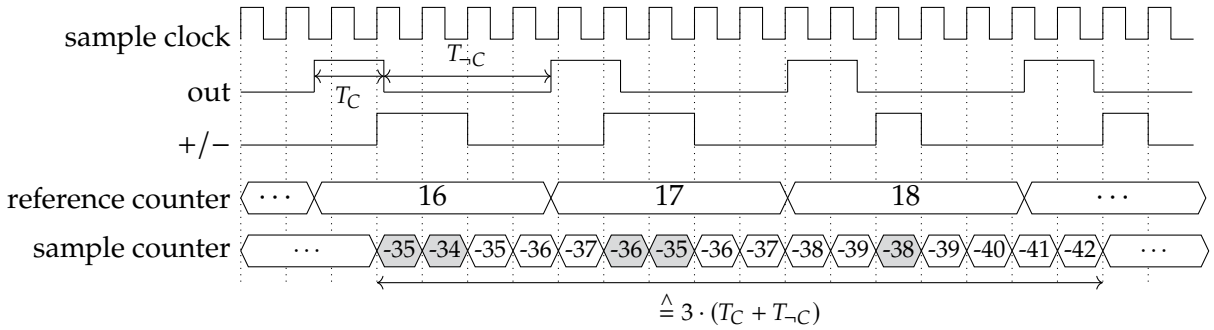



**Figure 5.8** Timing diagram of the ICLooPUF sample counter. ©2022 IEEE [21]

the Nyquist-Shannon theorem at the lower end and upper bounded by the available frequency of the chip, e.g., from the available PLL. Note that with decreasing $f_s$, the time to reach a reliable response increases, i.e., a higher sampling frequency is preferable. A counter counts with frequency $f_s$ in an alternating manner upwards and downwards for each one period of the oscillation of the ICLooPUF. For a period length $T$, the number of oscillations of the sampling clock in this period is $T \cdot f_s$, i.e., the counter value of the sampling counter is related to the period length of the ICLooPUF.

Conditioning the up and down count of the sample counter on the currently applied challenge and with challenge interleaving applied, the counter counts up for each $T_C$ and down for each $T_{\neg C}$. Consequently the counter value after any even number of oscillations of the ICLooPUF is related to the difference $T_C - T_{\neg C}$. The higher the sample clock $f_s$, the more precise is the resolution of $T_C - T_{\neg C}$.

The up and down counting in the measurement circuit as well as the enabling of the counter is controlled by the ICLooPUF. The T-FF at the ICLooPUF's output ensures that the output signal is logically 0 for exactly one period and logically 1 for the next period. This corresponds to the control signal for up and down counting. In addition, the ICLooPUF drives a reference counter. The system is stopped, when the reference counter reaches a pre-defined value $n_{max}$. This way, for all pairs of challenges and complementary challenges the same number of periods under $C$ and $\neg C$ is used to derive the delay difference. This approach is easy to implement and sample counter values are directly related to the difference in the ICLooPUF's period length for $C$ and $\neg C$.

The transition from the clock domain of the ICLooPUF oscillation to the clock domain of the sample clock can result in meta-stability. Therefore, the enable signal of the sample counter as well as the control signal for up and down counting are synchronized with the sample clock. Finally, the comparison result of the reference counter with $n_{max}$ is buffered to prevent wrong results in the sample clock domain.

The timing of the sample counter in relation to the ICLooPUF output signal is depicted in Fig. 5.8 for some exemplary ICLooPUF periods. The reference counter is increased every $T_C + T_{\neg C}$. The up/down signal (+/-) follows synchronized to the sampling clock with the delay of a two-stage synchronizer implemented by a cascade of two FFs. The sample counter counts up (highlighted in gray) when the up/down signal is high and down otherwise. Since $T_{\neg C} > T_C$ in the example, the counter value is overall decreasing in the depicted time interval.

**Quantization Error** In principle, a quantization error can occur in the suggested measurement circuit. Assuming jitter-free period lengths $T_C$ and $T_{\neg C}$ and $T_s = 1/f_s$ for the sampling clock, and the first rising edge of the sample clock occurring aligned with the ICLooPUF's first rising edge (but not triggering any action), the sampling counter value after $n_{max}$ periods of the output signal generated by the ICLooPUF is

$$\sum_{k=0}^{n_{max}-1} \lfloor [T_C + (k \cdot (T_C + T_{\neg C}) \bmod T_s)] \cdot f_s \rfloor - \lfloor [T_{\neg C} + (((k+1) \cdot T_C + k \cdot T_{\neg C}) \bmod T_s)] \cdot f_s \rfloor . \quad (5.15)$$

This equation is construed as: The up and down counting is periodic with $T_C + T_{\neg C}$; in each of these counting periods $C$ and $\neg C$ contribute to the number of increments and decrements with $T_C \cdot f_s$ and $T_{\neg C} \cdot f_s$. However, since $T_C$ and $T_{\neg C}$ are typically not multiples of $T_s$, a small amount of time expressed by the modulo terms is sampled as part of the wrong period. The quantization error is the difference of Eq. (5.15) and the expected outcome when $n_{max}$ is reached

$$(T_C - T_{\neg C}) \cdot n_{max} \cdot f_s. \tag{5.16}$$

Although the quantization error can get large in theory,[5] a small jitter in the clock frequency significantly reduces the quantization error. Therefore, although a quantization error must be considered in theory, it is negligible for practical implementations.

### 5.5.2 Theoretical Analysis of Side-Channel Attack Vectors

The ICLooPUF hides the value of the sample counter, corresponding to $T_C - T_{\neg C}$, from an attacker. This section provides a theoretical analysis of potential side-channel observations and concludes about the robustness of the protection principle. The following analysis considers jitter-free oscillations as the best case from an attacker's perspective.

#### Observation of Measurement Time

First, a possible timing side-channel of the ICLooPUF from observation of the runtime is considered. The attacker observes two alternating oscillations with unknown period lengths $T_C$ and $T_{\neg C}$. Further, the number of oscillations $n_{max}$ is not considered a secret, i.e., it is known by the attacker. Consequently, the measurement for a challenge pair $C$ and $\neg C$ takes $n_{max} \cdot (T_C + T_{\neg C})$ and the attacker can observe the sum $T_C + T_{\neg C}$ of the periods.

#### Observation of Oscillation Frequency

Second, the spectral side-channel of the ICLooPUF is investigated since for oscillation-based PUF primitives, the oscillation frequency is the most important attack vector. The oscillation of two interleaved challenges $C$ and $\neg C$ is modeled in the time domain as alternating sine waves without jitter and with period lengths of $T_C$ and $T_{\neg C}$ as

$$g(t) = g_1(t) + g_2(t) \tag{5.17}$$

with

$$g_1(t) = \sum_{k=1}^{N} \sin\left(\frac{2\pi}{T_C}\left(t - (k-1)\left(T_C + T_{\neg C}\right)\right)\right) \cdot \Theta\left(\frac{\left(t - \left((k-1)\left(T_C + T_{\neg C}\right) + \frac{T_C}{2}\right)\right)}{T_C}\right) \tag{5.18}$$

$$g_2(t) = \sum_{k=1}^{N-1} \sin\left(\frac{2\pi}{T_{\neg C}}\left(t - \left(kT_C + (k-1)\,T_{\neg C}\right)\right)\right) \cdot \Theta\left(\frac{\left(t - \left(k\left(T_C + T_{\neg C}\right)\right) + \frac{T_{\neg C}}{2}\right)}{T_{\neg C}}\right), \tag{5.19}$$

where $\Theta(t)$ is the rectangular function. For the signal modeled by Eqs. (5.17) to (5.19) at each point in time only one of the two oscillations is active, i.e., always one complete period of the oscillation under $C$ is alternated with one complete period under $\neg C$.

Following previous attacks on ring-based PUFs, the spectral amplitude is the attack vector for targeting the frequency. Transforming the time domain signal from Eq. (5.17) to the frequency domain (for details c.f. Appendix C) yields

[5] Both terms in Eq. (5.15) deviate from the best possible quantization value by utmost 1 for each $k$, so the quantization error is trivially bound by $2 \cdot n_{max}$.

$$|S(f)| = \left| \frac{(1+\gamma)^2 T_C n_{max}}{2\pi} \sum_{n=-N}^{N} \left[ \frac{1}{n^2 - (1+\gamma)^2} \left( e^{j2\pi \frac{n}{(1+\gamma)}} - 1 \right) + \frac{\gamma}{n^2\gamma^2 - (1+\gamma)^2} \left( 1 - e^{j2\pi \frac{n}{(1+\gamma)}} \right) \right] \cdot \operatorname{sinc}\left( n_{max}(1+\gamma) T_C \left( f - \frac{n}{(1+\gamma)T_C} \right) \right) \right| , \tag{5.20}$$

where $\gamma := T_{\neg C}/T_C,\ \gamma > 0$ is the ratio of period lengths. In Eq. (5.20), the global maxima of the sinc function are at multiples of the frequency defined by the sum of the periods $(1+\gamma)T_C = T_C + T_{\neg C}$. The width and the amplitude of the sinc functions are proportional to the sum $T_C + T_{\neg C}$ and the number of oscillations $n_{max}$. In other words – similar to observations of the measurement time – an attacker can observe the sum of the period lengths from the frequency spectrum.

#### Conclusion

From the measurement time and the frequency it is not possible to retrieve the secret $T_C - T_{\neg C}$ directly. For both attack vectors the sum $T_C + T_{\neg C}$ of the period lengths can be observed. However, under the assumption that single periods of $T_C$ and $T_{\neg C}$ cannot be resolved, the sum does not reveal information about its terms, nor about about their difference. Thus, the measurement time and the oscillation frequency of the ICLooPUF are no exploitable attack vectors.

### 5.5.3 Practical Implementation of the ICLooPUF

This section introduces an implementation of the ICLooPUF on a CW305 board that features an Artix-7 (XC7A100TFTG256) using a sample clock of $f_s$ =400 MHz. In accordance with Chapter 4, the design consists of 64 delay stages. Since challenges are Hadamard codewords like for the original Loop PUF and the all-zero and all-one challenges are dropped, this results in $N - 1 = 63$ bits derived from the PUF primitive.

#### Resource Utilization and Place-and-Route

The delay stages of the ICLooPUF are realized as LUTs, similar to Section 4.1.1. Fixed placement is only used for LUTs implementing delay stages and challenge interleaving as well as for the T-FFs used for challenge interleaving. In addition, input pins of the mentioned LUTs are fixed. The output T-FF of the ICLooPUF is fixed routed to ensure a short feedback. Apart from this, no fixed placement or routing is required in the circuit. In particular, routing between delay stages is left unconstrained since all connections between delay elements are part of the ring for every challenge and cancel out when comparing two frequencies. Hence, only the paths within delay elements can cause a bias that decreases the PUF quality. As for the original Loop PUF from Section 2.2.2 the use of Hadamard challenges ensures that bias related to path imbalance is compensated. Finally, four delay elements are grouped in a single slice for optimum resource usage.[6]

The challenge interleaving is implemented by 4-to-2 multiplexers realized in 6-input-2-output LUTs. These select from a pair of challenge bits $c_i, c_j$ and the respective complements $\neg c_i, \neg c_j$ either the two challenge bits or the two complementary challenge bits. Always two multiplexer-LUTs are implemented together with one T-FF triggering the switching of the challenges – implemented by a LUT and a FF – in one slice. This way a CLB[7] consists of one slice deriving four challenge bits and one slice with the corresponding four delay elements; the signal from the delay path to the clock input of the T-FF is decoupled through a latch in the slice with the delay elements. The regularity of the design does not only help to reduce bias in the PUF response – since always the same slice in a CLB is used

[6] Internal differences of the LUTs due to their position in a slice and on the FPGA might introduce slight bias. However, these effects are negligible for the design built to analyze side-channel protection.

[7] On Artix-7 FPGAs a CLB consists of two slices, where each slice features four LUTs and eight flip-flops.

for delay elements – but the existence of macro-blocks also supports a quick and easy design. With these design choices, the number of slices required for the ICLooPUF is twice the amount of slices needed for the original Loop PUF; area optimization at the cost of a more complicated placement and routing is possible.

#### Stability Considerations of Interleaving

In order to flip a challenge bit without inserting glitches while the PUF is oscillating, the same stable state must be applied at input and output of the delay elements. In other words, the delay of the feedback path from an inner node of the delay chain, through the T-FF to the multiplexer switching the challenge in addition to the delay from the challenge multiplexer input back to the delay element must be smaller than a half-period of the oscillation.

For the design the oscillation frequency is around 16 MHz corresponding to a half-period of 31.25 ns. Without dedicated place-and-route of the feedback, switching tuples of four delay stages of the 64 stages provides a sufficiently low delay. Increasing the number of stages that switch challenge bits in parallel would likely be possible, in particular if manual place-and-route would be applied for the feedback path. This would reduce resource allocation and increase the design effort but is considered out-of-scope for this work.

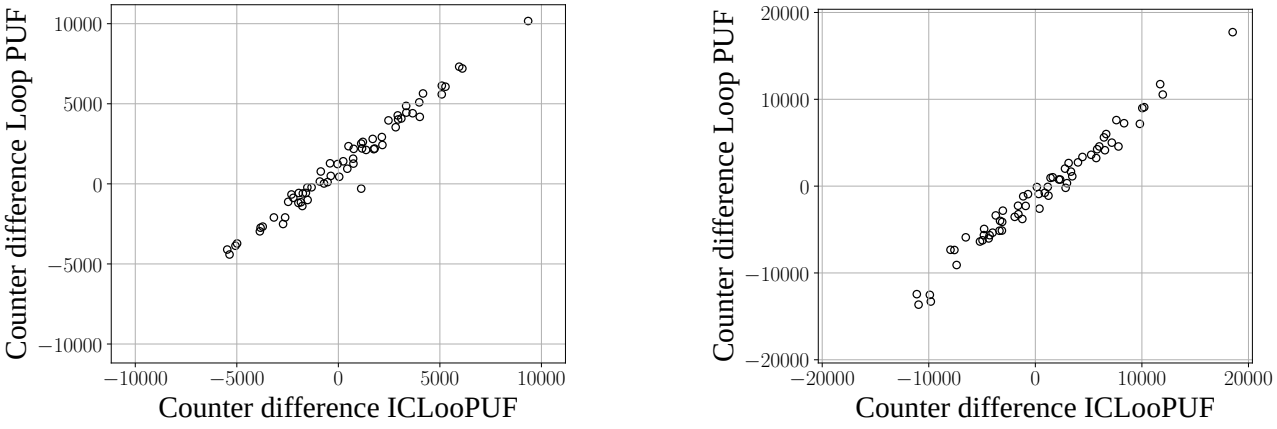
#### Offset Compensation

The oscillation of the ICLooPUF has a lower slew-rate compared to a normal clock signal. Combined with the specific propagation delay of the FPGA's internal gates, one (e.g., the rising) edge may propagate faster than the other (e.g., the falling) edge from the output T-FF of the ICLooPUF to the sample counter's input that selects up or down count. Consequently, the count of one half period of the oscillation at output T-FF is extended compared to the other half period, causing an offset of the sample counter value.

To compensate the device-specific offset, the measurement of the ICLooPUF is extended to two phases: First, $C$ and $\neg C$ are applied as interleaved challenges for $^{n_{max}}/_{2}$ periods. Neglecting noise, the sample counter value is $v_s^+ = v_d + v_o$, where $v_d$ is the actual difference and $v_o$ is the offset of the counter due to the asymmetric delay of the connect from the delay chain to the sample counter. Second, the challenges are exchanged, and $\neg C$ is applied as first challenge and $C$ as second challenge for another $^{n_{max}}/_{2}$ oscillation periods, such that the sample counter value is $v_s^- = -v_d + v_o$. The delay difference of oscillations under $C$ and $\neg C$ is computed from the difference $v_s^+ - v_s^- = 2 \cdot v_d$ on the device. Since the delay offset $v_o$ is independent of the order of the challenges and constant for the same device, it cancels out in the difference.

#### Extensions for Experiments

For the experiments the design is extended. First, it is possible to send arbitrary values of challenges $C$, and $\neg C$ to the device for interleaving; in an actual design, the challenges $C$ could be generated on-chip and $\neg C$ would be generated by inverting $C$ on-chip. Second, the design supports different modes of operation: An *interleaved* mode implements the challenge interleaving from Section 5.5.1; Two challenges are interleaved and the counter is counting down when the first challenge $C$ is applied and up for the second challenge $\neg C$. A *sequential* mode applies a single challenge $C$ or $\neg C$ without interleaving as for the original Loop PUF. The *sequential* mode is leveraged to verify the functionality of the *interleaved* mode in Section 5.5.4 and to determine the expected oscillation frequency of the ICLooPUF for SCA experiments in Section 5.5.5. Third, the reference value $n_{max}$ can be set for the experiments, and the design allows for reading out the value $v$ of the counter for PUF evaluations and as a reference for SCA.




**(a)** $n_{max} = 2^{17}$ **(b)** $n_{max} = 2^{18}$

**Figure 5.9** Comparison of counter value $v_{C-\neg C}$ averaged over 1000 repetitions for ICLooPUF and counter value difference $v_C - v_{\neg C}$ achieved without challenge interleaving (original Loop PUF) applying each the same challenges $\neg C$ and $C$. ©2022 IEEE [21]

### 5.5.4 Functional Validation of the ICLooPUF

This section evaluates the challenge interleaving of the ICLooPUF compared to sequential measurements of the original Loop PUF. Additionally, it provides a first PUF quality assessment, which shows possible trade-offs between reliability and runtime.

#### Equivalence of Interleaved and Sequential Mode

Fig. 5.9 shows the average counter values $v_{C-\neg C}$ for 1000 repetitions per challenge of the *interleaved* mode compared to the difference of averaged counter values $v_C - v_{\neg C}$ for $\neg C$ and $C$ in *sequential* mode, which is equivalent to the sequential operation of the original Loop PUF. The plots show a linear relationship, i.e., the two modes lead to the same results. Furthermore, doubling the reference counter values $n_{max}$ yields doubled counter values for both operation modes, which is the expected behavior due to the doubled measurement time. Concluding, the ICLooPUF is functionally equivalent to the original Loop PUF.

#### Preliminary PUF Quality Assessment

In the following, the sign-based bit derivation method from Section 2.3.1 and the TMHD from Section 2.3.2 are evaluated for the ICLooPUF. As the focus is on SCA hardening of the primitive, only a first assessment of the quality metrics is provided to validate that the ICLooPUF is in principle a valid PUF primitive. Note that an in-depth evaluation requires significantly more data and devices.

As the TMHD makes use of helper data, results for the sign-based method and *dark-bit masking* as described in Section 2.3.1 are provided as well, i.e., instead of all 63 bits only the $l$ most reliable bits are used. Neglecting specialized encoding schemes, storing the reliability information requires 63 bits of helper data as for TMHD, which enables a fair comparison of both methods. Table 5.8 provides results regarding the common PUF metrics *reliability*, *uniqueness*, and *uniformity* – computed as in [102] – for varying values of the reference counter value $n_{max}$. The *reliability* illustrates the reproducibility of a PUF response and is defined via the intra-device Hamming distance of a response $\mathbf{r}_{t,d}$ with respect to the reference response $\mathbf{r}_{enroll,d}$. For $D$ different devices, the average reliability is

$$\text{Reliability} = \frac{1}{D}\sum_{d=1}^{D}\left(1 - \frac{1}{R}\sum_{t=1}^{R}\frac{\text{HD}(\mathbf{r}_{enroll,d}, \mathbf{r}_{t,d})}{|\mathbf{r}|}\right),$$

**Table 5.8** PUF metrics of the ICLooPUF for ten devices, $l$ bits, and 100 repetitions (from which ten are taken for enrollment). ©2022 IEEE [21]

| | | | $n_{max}$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | $2^{10}$ | $2^{11}$ | $2^{12}$ | $2^{13}$ | $2^{14}$ | $2^{15}$ | $2^{16}$ | $2^{17}$ | $2^{18}$ |
| Reliability | Sign | 63 | 0.797 | 0.834 | 0.880 | 0.909 | 0.935 | 0.956 | 0.969 | 0.978 | 0.983 |
| | Sign | 60 | 0.809 | 0.850 | 0.895 | 0.925 | 0.953 | 0.972 | 0.984 | 0.991 | 0.996 |
| | Sign | 50 | 0.850 | 0.893 | 0.935 | 0.965 | 0.982 | 0.992 | 0.998 | 1.000 | 1.000 |
| | Sign | 32 | 0.912 | 0.951 | 0.979 | 0.991 | 0.997 | 1.000 | 1.000 | 1.000 | 1.000 |
| | TMHD | 63 | 0.782 | 0.846 | 0.910 | 0.952 | 0.978 | 0.993 | 0.998 | 1.000 | 1.000 |
| Uniformity | Sign | 63 | 0.508 | 0.510 | 0.514 | 0.522 | 0.525 | 0.529 | 0.525 | 0.525 | 0.525 |
| | TMHD | 63 | 0.479 | 0.479 | 0.475 | 0.471 | 0.475 | 0.470 | 0.476 | 0.481 | 0.470 |
| Uniqueness | Sign | 63 | 0.482 | 0.481 | 0.485 | 0.480 | 0.482 | 0.474 | 0.472 | 0.479 | 0.476 |
| | TMHD | 63 | 0.487 | 0.499 | 0.504 | 0.494 | 0.500 | 0.490 | 0.493 | 0.493 | 0.486 |

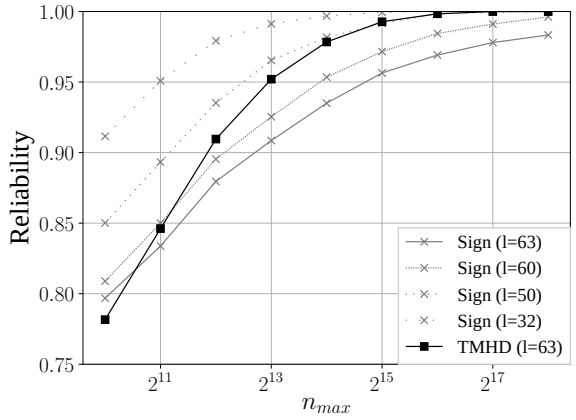



**Figure 5.10** Reliability of the ICLooPUF with increasing $n_{max}$ for ten devices and 100 repetitions per challenge; ten repetitions are taken to derive the expected responses (enrollment). Adapted from ©2022 IEEE [21].

where $R$ is the number of repetitions and $|\mathbf{r}|$ is the number of bits of the PUF response. Ideally, the reliability takes a value of 1. Note that sometimes the average BER computed as BER = 1 − Reliability is provided instead.[8] The *uniqueness* provides a metric for the difference of PUF responses on different devices by

$$\text{Uniqueness} = \frac{2}{D(D-1)} \sum_{d=1}^{D-1} \sum_{d'=d+1}^{D} \frac{\text{HD}(\mathbf{r}_d, \mathbf{r}_{d'})}{|\mathbf{r}|},$$

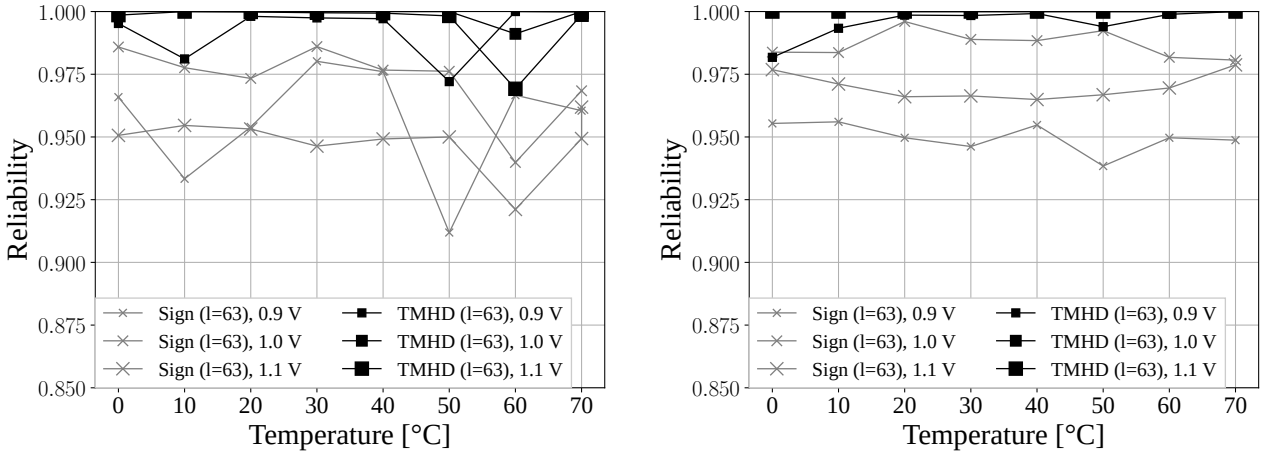
where a value of 0.5 is ideal. Finally, the *uniformity* estimates the relative frequencies of bits in the PUF response. For $D$ different devices the average uniformity is

$$\text{Uniformity} = \frac{1}{D} \sum_{d=1}^{D} \frac{1}{|\mathbf{r}|} \sum_{i=1}^{|\mathbf{r}|} r_{i,d},$$

where again a value of 0.5, i.e., a uniform distribution of 0's and 1's, is considered ideal. In Table 5.8, as expected, uniformity and uniqueness are independent of the reference counter value $n_{max}$.[9] Both metrics are close to their optimal values of 0.5.

[8] The BER allows designers of ECCs to determine the required correction capability, e.g., from Table 5.1 for $64 \cdot 100 \cdot 10$ samples a reliability of 1 would result in a average BER of $\leq 1.5 \cdot 10^{-5}$.

[9] The number of bits $l$ does not have an effect either, therefore the respective rows are omitted in Table 5.8.




**(a)** $n_{max} = 2^{17}$ **(b)** $n_{max} = 2^{18}$

**Figure 5.11** Reliability of the ICLooPUF with varying environmental temperature and supply voltage for a single device and 100 repetitions per challenge; ten repetitions are taken to derive the expected responses at 20 °C and 1 V (enrollment). Adapted from ©2022 IEEE [21].

Regarding reliability, Fig. 5.10 depicts the values from Table 5.8 for an easier comparison. The results provide further motivation to use the TMHD approach: for $n_{max} \geq 2^{17}$ the method leads to nearly perfectly reliable reconstruction under nominal conditions. On the other hand, using the sign-based method a similar reliability can be achieved for taking only the $l = 50$ most reliable bits, i.e., deriving 13 bits less than for the TMHD. Note that the runtime to derive the PUF response is proportional to $n_{max}$, i.e., the TMHD provides the best trade-off regarding runtime and derived bits, whereas for the sign-based derivation either a significant loss of bits has to be tolerated or the runtime has to be increased to guarantee a stable response. The reliability results for the ICLooPUF and sign-based reconstruction are comparable to the results for Loop PUF primitives from [6], where different oscillator-based PUF primitives are implemented on Artix-7 FPGAs.[10] Using the TMHD improves the reliability even further.

Fig. 5.11 shows the reliability for a selected device under varying environmental conditions $n_{max} = 2^{17}$ (Fig. 5.11a) and $n_{max} = 2^{18}$ (Fig. 5.11b). Differential measurements of $C$ and $\neg C$ help to compensate parts of the disturbance from environmental changes since the oscillators for each challenge are expected to be affected similarly. Similarly as for the Loop PUF [6], for the nominal supply voltage of 1 V (medium size markers), the ICLooPUF is stable for TMHD and sign-based bit derivation within the temperature range from 0 °C to 70 °C. Varying the supply voltage by ±10% decreases the reliability in particular for the sign-based method as delay values are individually affected by the supply voltage and the differential measurements compensate only a part of the effects. In general, the TMHD is less affected by the environmental conditions and retains reliabilities above 95% ($n_{max} = 2^{17}$) respectively 98% ($n_{max} = 2^{18}$) even under extreme conditions.

Summing up, the proof-of-concept design of the ICLooPUF is a highly reliable PUF primitive. The use of the TMHD improves the number of extracted bits for a targeted reliability and runtime compared to the sign-based method, and is more robust regarding variations of the environmental conditions.

### 5.5.5 Validation Against Side-Channel Analysis

This section provides a practical validation of the side-channel protection of the ICLooPUF. The power side-channel over the CW305 board's shunt resistor is measured as, according to Section 4.1.4,

[10] Reliability results in [6] are generated with 1.3 ms measurement time, which corresponds to $n_{max} = 2^{14} - 2^{15}$ for the presented ICLooPUF.

EM measurements do not provide additional benefit for attacks on the Loop PUF. The time domain measurements acquired by a PicoScope 6402 USB oscilloscope at sampling frequency $f_s$ =156.25 MS/s are transformed into the frequency domain. In accordance with the results from Section 5.5.4 the reference counter value is set to $n_{max} = 2^{18}$; this results in a high reliability and the attacker is able accumulate more information compared to lower values of $n_{max}$.

#### Exploration of Frequencies of Interest

The original Loop PUF leaks by the oscillation frequency. Considering in addition Eq. (5.20), the dependency between the frequency difference of the PUF with $C$ and $\neg C$ applied on the one hand and the amplitude, width and frequency of the ICLooPUF's spectrum on the other hand has to be investigated. In order to enable the analysis, the *sequential* mode described in Section 5.5.3 allows to determine the FoI range. For each challenge $C$, the counter value $v_C$ is acquired and the frequency of the delay chain is

$$f_{oI} = \frac{2 \cdot n_{max}}{v_C \cdot T_{ref}}. \tag{5.21}$$

Note that due to the output T-FF, which acts as a frequency divider, the reference counter measures only half of the frequency and the factor of two has to be added to determine the oscillation frequency of the loop. The average frequency across all challenges $C_i$ and $\neg C_i, i \in \{1, \ldots, 63\}$ is $\bar{f}_{oI}$ = 16.021 MHz with a frequency range 16.007 MHz $\leq f_{oI} \leq$ 16.035 MHz, which defines the region of interest for the following SCA evaluation.

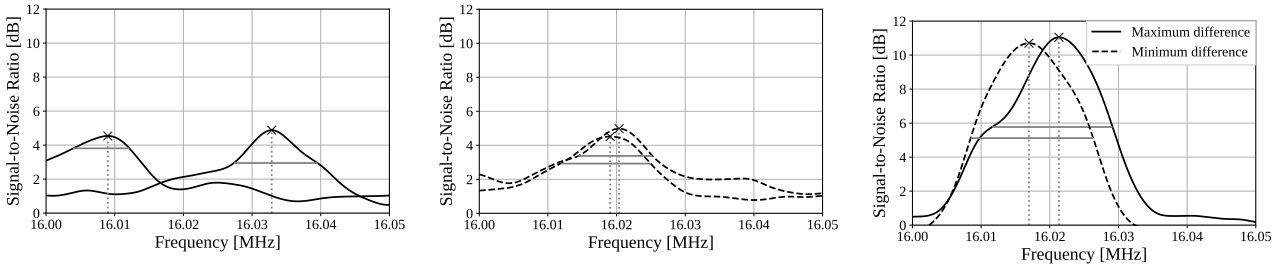
#### Side-Channel Analysis Results

Considering the average frequency of $\bar{f}_{oI}$ = 16.021 MHz and the value $n_{max} = 2^{18}$, the expected runtime of the ICLooPUF per challenge is around 32.7 ms, from which the first 30 ms are transformed into the frequency domain.[11] Finally, for easier peak detection, the frequency spectrum is low-pass filtered, similar to Section 4.1.3.

Fig. 5.12 shows exemplary spectra in the FoI range from 16 MHz to 16.05 MHz. Figs. 5.12a and 5.12b correspond to challenges $C$ and $\neg C$ with the maximum and minimum counter difference in *sequential* mode, i.e., the extreme values an attacker can observe for the original Loop PUF. In Fig. 5.12c the spectra for the same challenges in *interleaved* mode are depicted, where the solid line corresponds to Fig. 5.12a, and the dashed line corresponds to Fig. 5.12b. Note that the increased amplitude in Fig. 5.12c compared to Figs. 5.12a and 5.12b stems from the fact that only in *interleaved* mode the challenges are switched by T-FFs, while in *sequential* mode the T-FFs are deactivated, i.e., the increased amplitude is caused by the additional switching activity. The difference of frequencies in the spectra in Figs. 5.12a and 5.12b corresponds to the underlying counter differences, i.e., an attacker can learn from the peak comparison. In Fig. 5.12c, the depicted extreme cases show similar spectra corresponding to the *averaged* frequency of the ICLooPUF challenged with $C$ and $\neg C$ interleaved. At first glance, there is no obvious attack vector on the ICLooPUF's spectrum visible.

In order to further evaluate the side-channel resistance of the ICLooPUF, the real counter values and their amplitudes are compared with different properties of the observed frequency peaks. As mentioned above, the amplitude, the frequency and the width of the peak are considered, which are marked in Fig. 5.12 as cross, dotted gray vertical line and solid gray horizontal line respectively. Pearson's correlation coefficient of the counter values with the characteristics of the peak is used as a measure for similarity. The correlation represents the predictability of the counter values from SCA. In Table 5.9 the correlations between peak characteristics and counter values are provided for the original Loop PUF as well as for the ICLooPUF. The lower part of the table provides correlations of

[11] Additionally, a noise floor is determined by connecting the system clock instead of the delay chain to the counting circuitry. Similar to Sections 3.1.4 and 4.1.3, subtracting the noise floor from the actual signal allows to remove regular components of the device, such as the clock frequency, for better detecting relevant frequencies, but is not a necessary condition for the attack.




**(a)** *sequential* mode, $\max_i(v_{C_i} - v_{\neg C_i})$ **(b)** *sequential* mode, $\min_i(v_{C_i} - v_{\neg C_i})$ **(c)** *interleaved*, $v_{C_i - \neg C_i}$

**Figure 5.12** Frequency spectra revealed from SCA in the FoI range for extreme cases of expected frequencies. Challenges $C_i$ and $\neg C_i$ with extreme cases (a)-(b) in *sequential* mode, and (c) the corresponding spectra in *interleaved* mode. ©2022 IEEE [21]

**Table 5.9** Side-channel analysis results: correlation of counter values and physical observations averaged over different numbers of measurement traces and for $n_{max} = 2^{18}$ and $N - 1 = 63$. ©2022 IEEE [21]

| | Traces | Frequency | Amplitude | Width |
|---|---|---|---|---|
| Loop PUF ($C$) | 1 | -0.961 | 0.122 | -0.115 |
| Loop PUF ($C$) | 10 | -0.995 | -0.213 | -0.309 |
| Loop PUF ($\neg C$) | 1 | -0.970 | -0.103 | 0.089 |
| Loop PUF ($\neg C$) | 10 | -0.987 | 0.018 | -0.109 |
| ICLooPUF ($\delta_i$) | 1 | 0.173 | -0.135 | -0.003 |
| ICLooPUF ($\delta_i$) | 10 | 0.156 | -0.073 | -0.028 |
| ICLooPUF ($\delta_i$) | 100 | 0.233 | -0.104 | -0.067 |
| ICLooPUF ($\lvert\delta_i\rvert$) | 1 | -0.038 | 0.041 | 0.202 |
| ICLooPUF ($\lvert\delta_i\rvert$) | 10 | 0.098 | 0.093 | 0.409 |
| ICLooPUF ($\lvert\delta_i\rvert$) | 100 | 0.181 | 0.128 | 0.406 |

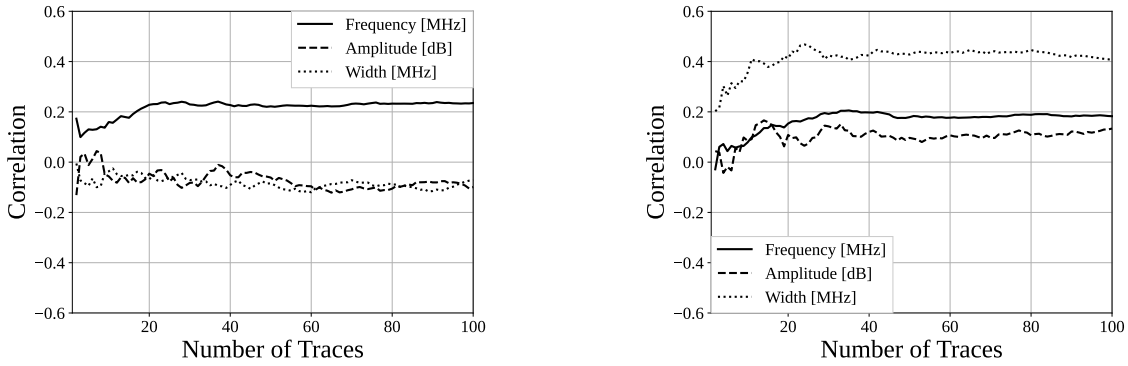
the absolute counter values to analyze the impact on amplitude-based bit derivation. An attacker could try to aggregate information from several observations by using repeated measurements of the same challenge. Therefore, in Table 5.9 the peak characteristics are determined from the average of several traces of the same challenge and compared to *averaged counter values*.

Even with a single measurement per challenge in *sequential* mode the match between the frequency and the counter value leads to an absolute correlation of above 0.961,[12] i.e., as expected there is a direct relationship between frequency and counter value for the original Loop PUF. Adding further measurements increases the correlation. For the original Loop PUF, the observed frequencies of $C$ and $\neg C$ therefore reveal sign and amplitude of the frequency difference, and hence the secret as shown in Sections 4.1.4 and 5.2.5.

In Table 5.9, frequency, amplitude, and width show correlations of up to 0.409 with the counter values and the absolute counter value $|\delta_i|$ for the ICLooPUF.[13] A detailed insight of the evolution of the correlation is provided in Fig. 5.13, where Figs. 5.13a and 5.13b depict the correlation with increasing number of averaged traces per challenge for the counter values respectively their absolute value. In Fig. 5.13a, the correlations are below 0.25. Similarly, in Fig. 5.13b correlations with frequency and amplitude converge towards values of below 0.2 with increasing measurements, and the width to around 0.4. In other words increasing the number of repetitions beyond the depicted number does not improve the match of observation and counter values for the ICLooPUF.

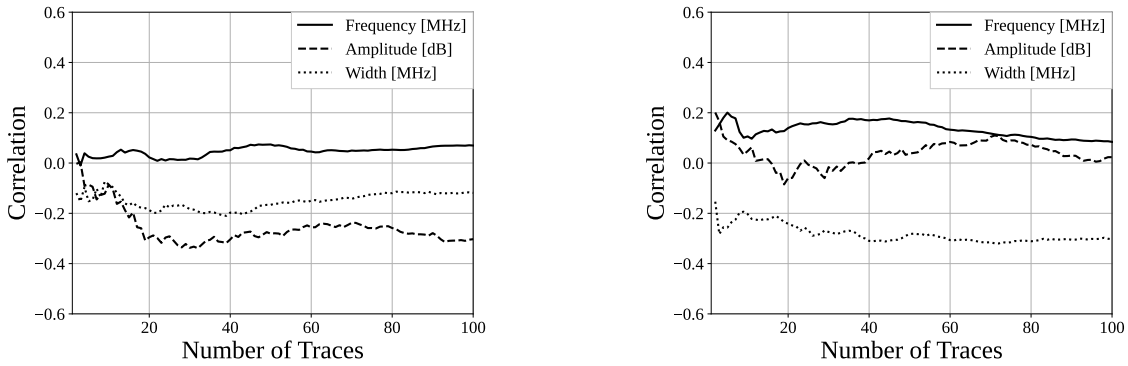
[12] For $N = 63$ observations, the confidence interval of the correlation coefficient is within $[0.9361, 0.9763]$ with a confidence level of 0.95.

[13] The confidence interval of the correlation coefficient is within $[0.1794, 0.5963]$ with a confidence level of 0.95.




**(a)** correlation with $\delta$ **(b)** correlation with $|\delta|$

**Figure 5.13** Evolution of correlation between averaged counter values and observed side-channel properties for 63 challenges of the ICLooPUF, same device as used for Table 5.9. ©2022 IEEE [21]




**(a)** correlation with $\delta$ **(b)** correlation with $|\delta|$

**Figure 5.14** Evolution of correlation between averaged counter values and observed side-channel properties for 63 challenges of the ICLooPUF, different device as used for Table 5.9. ©2022 IEEE [21]

Even though correlation values of 0.4 do not indicate a causal relation with the counter values, Fig. 5.14 investigates whether the same correlation is observed on a different device to rule out any profiling attacks. On a second device the maximum absolute correlation is 0.31,[14] but the sign differs compared to Fig. 5.13. Therefore, even if correlations would theoretically allow for reducing entropy on a particular device – which is not indicated by the measured correlations – an attacker could not derive the device-specific correlation from profiling on a second board. Finally note that, e.g., for an attack on the TMHD scheme, the intervals for bit derivation would be wrongly estimated if there is no deterministic relation between observations and counter values, i.e., the required precision for an attack is not given by weak correlations.

Summing up, the ICLooPUF does not leak exploitable side-channel information in the frequency domain via frequency, amplitude or width of peak in the FoI range. Therefore, it constitutes an SCA-hardened PUF primitive.

[14]The confidence interval of the correlation coefficient is within $[0.0674, 0.5180]$ with a confidence level of 0.95.

# 6 Comparison of Attacks and Countermeasures for Ring-Based PUF Primitives

This chapter brings together the findings from Chapters 3 to 5 about SCA of ring-based PUF primitives and its prevention by suitable countermeasures. First, Section 6.1 summarizes the attacks on ring-based primitives in this work and relates them to attacks in other works. Section 6.2 revisits the proposed countermeasures for the Loop PUF and provides a comparison regarding their implementation cost and protection level for different bit derivation schemes. Furthermore, the transfer of the proposed countermeasures to the RO PUF is explored, and possible countermeasures for the TERO PUF are outlined. Finally, Section 6.3 concludes the chapter.

Fig. 6.1 revisits the overview of ring-based PUF primitives from Fig. 2.3a, and shows attacks (ϟ) and countermeasures (🛡) for the different primitives.[1] The work from this thesis is depicted by black symbols, while other works are highlighted in gray.

## 6.1 SCA Attacks on Ring-Based PUF Primitives

As shown in Section 2.5.2, most previous works target the RO PUF that leaks through its frequency emanations [56, 57, 58, 59]. The SCA attacks on the Loop PUF from Chapters 4 and 5 further emphasize that the oscillation frequency of ring-based PUF primitives can be exploited. Compared to the RO PUF, the Loop PUF is built from a longer ring with more stages resulting in a lower oscillation frequency. The lower frequencies can even be attacked by remote SCA as shown in Section 4.2. Further, the evaluation principle with sequential measurements results in a temporal separation of the leakage. This eases attacks on the Loop PUF compared to RO PUFs that run several oscillations in parallel, i.e., where it is necessary to resolve the oscillators, respectively the counters, locally.

The SCA of the TERO PUF in Chapter 3 highlights that metastable oscillations used for building PUF primitives are prone to attacks as well. However, the estimation of the oscillation duration is more challenging compared to measuring the frequency, which is also highlighted by other work that targets the TERO PUF [65].

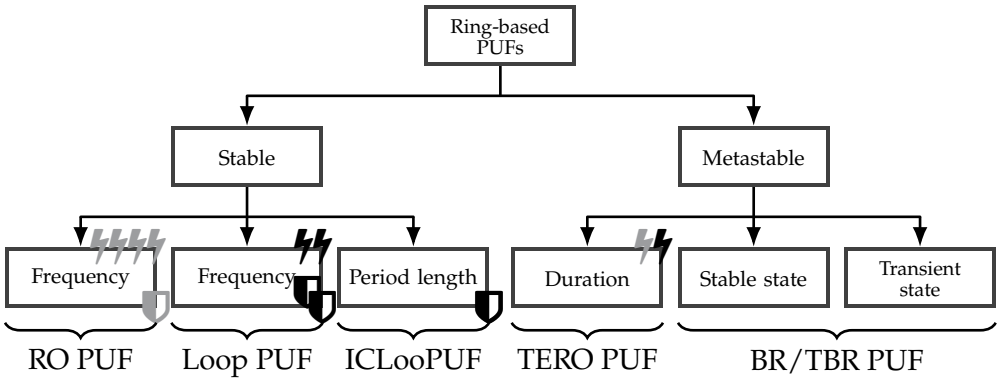



**Figure 6.1** Attacks (ϟ) and countermeasures (🛡) for ring-based PUF primitives. Black symbols: this thesis. Gray symbols: other works.

[1] In order to distinguish the RO PUF and the Loop PUF primitives, different from Fig. 2.3a, the evaluation of the frequency is split into two distinct blocks.

**Table 6.1** Comparison of countermeasures for an $N$-stage Loop PUF regarding protection level and required resources. Adapted from ©2022 IEEE [21].

| | Temporal Masking (Section 4.3.1) | Challenge Randomization (Section 5.3) | ICLooPUF (Section 5.5) |
|---|---|---|---|
| Hardware Overhead | very low | high | low |
| Timing | deterministic | probabilistic | deterministic |
| Randomness Source | Loop PUF | TRNG | not needed |
| Random Bits | $N-1$ | $\geq (N-1)\lceil \log_2(N-1) \rceil$ | 0 |
| Sign-Based | full | full | full |
| Two-Metric Helper Data | none | full | full |
| Equiprobable Quantization | sign only | full | full |
| Order Encoding | sign only | full | full |

From Fig. 6.1, the BR PUF and TBR PUF have not been subject to SCA so far. In theory, the methods used for the other PUF primitives can be transferred to measure the frequency and duration. However, due to the evaluation principle based on the stable or transient state, the secret is not directly related to the frequency or duration of the oscillation. A possible attack vector is to resolve the toggling of a single stage of the BR. The required localization and precision of the measurements for such an attack is very high though, and the practical feasibility needs further investigation.

In conclusion, for unprotected primitives the order of difficulty for an SCA attack, starting with the easiest, is: Loop PUF, RO PUF, TERO PUF, and BR/TBR PUF. Primitives that generate the PUF secret from the stable frequency of a ring are particularly prone to SCA as the frequency spectrum can be easily evaluated. On the other hand, if the secret is not directly related to the frequency, the complexity of the measurement methods and their required precision for SCA increases significantly.

## 6.2 Countermeasures for Ring-Based PUF Primitives

This section summarizes the proposed countermeasures for the Loop PUF. Further, a possible transfer of the Loop PUF countermeasures to other ring-based PUF primitives is explored. First, Section 6.2.1 compares resource requirements and the protection level of countermeasures for the Loop PUF. Second, Section 6.2.2 applies the temporal masking countermeasure to the RO PUF. Finally, Section 6.2.3 shows that, due to architectural similarities, the TERO PUF can be protected by countermeasures for the RO PUF.

### 6.2.1 Protection Mechanisms for the Loop PUF

In Chapters 4 and 5, different countermeasures have been proposed to protect the Loop PUF: *temporal masking* in Section 4.3.1, *challenge randomization* in Section 5.3, and *challenge interleaving* with the ICLooPUF in Section 5.5. Table 6.1 provides a comparison of the countermeasures regarding the required resources and the protection level for different bit derivation schemes.

**Resource Requirements** The temporal masking countermeasure, as sketched in Fig. 4.14, adds only $N+1$ XOR gates and a 1-bit storage to the original Loop PUF, resulting in a very low overhead. For each derived PUF bit $r_C$, one bit of randomness determines the order of the challenges $C$ and $\neg C$. However, a dedicated TRNG is not required as the bits are derived from the counter LSB, i.e., from the jitter of the Loop PUF frequency.

The challenge randomization approach needs to implement an algorithm as depicted in Fig. 5.5 and requires a TRNG, resulting in a significant hardware overhead. The countermeasure has a

probabilistic runtime due to possible re-sampling of duplicate indices, and requires additional time for randomizing the challenge order. In addition, the number of required random bits is significant, which hinders applications for low-cost embedded devices.

According to Table 5.7 and Fig. 5.7, the ICLooPUF requires $N$ additional 2-bit multiplexers and utmost $N$ T-FFs for switching challenge bits; in the proof-of-concept design in Section 5.5.3 $N/4$ T-FFs have been used. Note that the 2-bit multiplexers can be replaced by XOR gates inverting the challenge $C$ for every second period, further reducing the hardware footprint. Different from the Loop PUF, the ICLooPUF uses time domain sampling with an up/down counter. Although the counters and the comparator are arranged differently, the only overhead for the measurement circuit is the synchronization stage consisting of few FFs, which is negligible. The protection mechanism has therefore overall a low hardware overhead when compared to the original Loop PUF and provides resilience against SCA without requiring any random bits.

**Protection Level** Temporal masking can only protect the sign-based bit derivation. As discussed in Section 5.1, it has limited effect in the case of EPQ and order encoding or no effect in the case of TMHD. On the other hand, challenge randomization and the ICLooPUF protect against SCA with sign-based as well as with amplitude-based bit derivation.

Overall, if the reduced reliability of the sign-based bit derivation can be tolerated at the cost of error correction or reduced entropy, temporal masking has the lowest resource overhead. However, in order to enable highly efficient and reliable bit derivation, amplitude-based schemes are needed. The ICLooPUF maintains a low hardware overhead and prevents the need for a separate TRNG compared to challenge randomization when using these schemes.

### 6.2.2 Transfer of Loop PUF Countermeasures to the RO PUF

For the RO PUF from Section 2.2.1, several countermeasures have been proposed: (*i*) Restricting to non-overlapping pair-wise comparisons, i.e., each oscillator is used only to derive a single bit [57]. (*ii*) Increasing the number of counters to decrease the attacker's SNR as more rings oscillate simultaneously [57]. (*iii*) Interleaved placement of counters and multiplexers to prevent an attacker from spatially resolving them [58]. (*iv*) Measurement path randomization to break the relation of RO and attacked measurement path, where the path from the RO through the multiplexer to the counter is altered for each measurement [58].

Note that the adjacent placement of counters is vulnerable to attacks that exploit geometrical leakage, i.e., the fact that depending on the distance to the counter, the signal amplitude varies [59]. However, the ROs itself have not been spatially resolved so far, which is the underlying assumption of all countermeasures for the RO PUF.

Measurement path randomization comes at the cost of a significant overhead as the size of the required multiplexers increases, and a TRNG is required to randomize the address of the selected counter [58]. Thus, the question arises whether it is possible to reduce the overhead by transferring a countermeasure for the Loop PUF, which uses sequential measurement of frequencies by a single counter, to the RO PUF, which is based on parallel frequency measurements of two ROs connected to separate counters. In theory, challenge randomization could be applied to the RO PUF, i.e., the order of compared RO pairs would be randomized. However, the same drawbacks in terms of the hardware overhead apply as for the Loop PUF and there is no benefit compared to measurement path randomization. As challenge interleaving implemented by the ICLooPUF is limited to a configurable ring, it is not applicable.

The temporal masking countermeasure can be applied if the frequencies of two selected ROs are measured sequentially by the same counter, as for the Loop PUF. It renders attacks infeasible that spatially resolve the counters [58, 59], as long as the ROs themselves cannot be spatially resolved. As multiple RO pairs are measured to derive a sufficient number of bits from the RO PUF, different design trade-offs are possible: *(i)* For sequential measurements using the same counter, the latency

to get the PUF response is doubled while the number of counters is halved. *(ii)* Keeping the number of counters constant allows measurements of the same amount of ROs in parallel as in the classical RO PUF design. Note that in order to avoid side-channel leakage, the measured ROs must belong to different RO pairs. Otherwise the same attacks as for classical RO PUFs would be possible. As only the way of parallelization is changed, the latency stays the same in this second case. Additional resources are needed if no up/down counters are used, e.g., in form of additional memory to cache measured frequencies and to store required random bits for the first activated ROs.

Summarizing, in terms of complexity, the number of counters can be reduced down to a single counter. However, the area required for the large number of ROs in a typical RO PUF design is much larger than the area of a single counter. Hence, the number of counters is not limited by area constraints but rather by the latency requirement. Contrary to the path randomization method [58], there is no specific design effort required for the protection.

### 6.2.3 Countermeasures for the TERO PUF

The SCA attacks on the TERO PUF in Chapter 3 highlight that the primitive needs countermeasures for a secure operation. Due to the similar architectures, the TERO PUF can be protected by the same mechanisms as the RO PUF: (*i*) Increasing the number of counters decreases the attacker's SNR as more TEROs oscillate simultaneously. (*ii*) Dedicated placement of counters and multiplexers increases the difficulty of resolving them spatially, but does not prevent sophisticated geometric leak attacks completely [59]. (*iii*) Restricting the TERO PUF to non-overlapping pair-wise comparisons, i.e., a cell is only used once, and limiting to sign-based bit derivation impedes the attacks from Section 3.3 entirely.

Following Section 6.2.2, the application of temporal masking similar as for the RO PUF, and alternatively to (*iii*) is possible. If the TERO cells are measured sequentially, comparisons can be made on the stored counter values. Hence, the oscillation of a single cell can only be observed once revealing no additional side-channel leakage. Additionally, shuffling of compared TERO cells impedes averaging of measurements with the same TERO cell as the attacker does not know to which cell pair a measurement belongs. Similar to challenge randomization for the Loop PUF, a significant overhead would be required.

## 6.3 Conclusion

Summing up, the Loop PUF's disadvantage of a single counter, which eases SCA by temporal separation of the leakage, turns out to be an advantage for the protection against SCA. For sign-based bit derivation temporal masking has a very low resource overhead. It can be transferred to multi-ring array PUFs such as the RO PUF and TERO PUF, providing an alternative to more expensive countermeasures as measurement path randomization. For amplitude-based bit derivation the ICLooPUF protects the Loop PUF with low resource overhead and without requiring any randomness. As the ICLooPUF is limited to configurable single-ring PUFs, there are no comparable countermeasures for the RO PUF and TERO PUF.

# 7 Side-Channel Analysis of BCH Codes

This chapter investigates SCA attacks on the error correction in Fig. 2.1. The focus is on BCH codes that are widely used to correct errors in the PUF response to enable a reliable key generation as outlined in Section 2.4. The data processed by the BCH code is related to the secret key, and an SCA attack that retrieves the codeword can determine the key as well. In Section 7.1 the required theoretical background of BCH codes is introduced, followed by their hardware implementation in Section 7.2. Section 7.3 introduces existing DPA attacks on BCH codes used in concatenation with a repetition code. Next, Section 7.4 extends the existing attacks by including the syndrome calculation to the power model, and altering the manipulation of helper data. As a result, the improved attack does not rely on the concatenation with a repetition code. Furthermore, a horizontal SCA attack is outlined that combines the power consumption of different points in time from a single trace. The attack is applicable even if the helper data is not altered, reducing the requirements compared to vertical SCA attacks, such as DPA. Finally, Section 7.5 provides results from simulated side-channel measurements for the vertical and the horizontal attacks, and Section 7.6 concludes the chapter.

This chapter combines and extends results from two master's theses that have been supervised as part of this thesis. The code generation framework from *Huber: "Design and Side-Channel Evaluation of BCH Code Hardware Implementations", Technical University of Munich, May 2022* [103] is used to generate simulated power traces of different BCH codes, and the horizontal attack from *Brosch: "Horizontal Side-Channel Analysis of Error-Correcting Codes", Technical University of Munich, November 2019* [104] is evaluated on the different implementations.

## 7.1 Theoretical Background of BCH Codes

Bose-Chaudhuri-Hocquenghem (BCH) codes are linear cyclic block codes that encode a $k$-bit message block $\mathbf{u}$ into an $n$-bit codeword $\mathbf{c}$, where $n > k$. After transmission over a noisy channel, the noisy codeword $\mathbf{c}' = \mathbf{c} \oplus \mathbf{e}$ is decoded to the received message $\hat{\mathbf{u}}$. In the following, the most important subclass of *binary* BCH codes is considered, i.e., the message and the codeword are binary vectors. The most relevant aspects of BCH codes are derived from [45], where further details and derivations can be found.

**Linear Block Codes** A block code is linear if the sum of two codewords $\mathbf{c}_1$ and $\mathbf{c}_2$ is also a valid codeword $\mathbf{c} = \mathbf{c}_1 \oplus \mathbf{c}_2$. Linear block codes can be expressed in terms of their *generator matrix* $\mathbf{G}$ of size $n \times k$ and with binary entries. For *systematic* linear block codes, the message and redundancy part of the codeword are separated into two parts and the generator matrix consists of a $k \times k$ identity matrix $\mathbf{I}$ and an $(n-k) \times k$ parity matrix $\mathbf{P}$, such that

$$\mathbf{c} = \mathbf{G}\mathbf{u} = \begin{bmatrix} \mathbf{I} \\ \mathbf{P} \end{bmatrix} \mathbf{u}, \tag{7.1}$$

where $\mathbf{u}$ is a column vector. From Eq. (7.1) all $2^k$ possible codewords can be generated. The *parity check matrix* $\mathbf{H}$ allows for verifying whether a codeword has been generated by the generator matrix $\mathbf{G}$. The columns of the $n \times (n-k)$ parity check matrix are orthogonal to the columns of $\mathbf{G}$, i.e., only for

valid codewords the syndrome is $\mathbf{H}^T\mathbf{c} = \mathbf{0}$. For a received codeword that is subject to an error $\mathbf{e}$, the syndrome

$$\mathbf{s} = \mathbf{H}^T\mathbf{c}' = \mathbf{H}^T(\mathbf{c} + \mathbf{e}) = \mathbf{H}^T\mathbf{e} \tag{7.2}$$

is non-zero and allows for detecting and correcting errors. The number of errors $t$ that can be corrected by a linear code depends on the *minimum distance* $d_{min}$, i.e., the minimal Hamming distance between any codewords of the code, and is bound by

$$d_{min} \geq 2t + 1. \tag{7.3}$$

A code of length $n$ with a message size $k$ that is capable of correcting $t$ errors in a received codeword is denoted as $(n, k, t)$ code.

**Cyclic Codes** Cyclic codes are an important subclass of linear block codes with an algebraic structure that allows for efficient decoding methods. A codeword $\mathbf{c}$ of a cyclic code that is shifted by $n'$ mod $n \geq 1$ positions to the right is again a valid codeword $\mathbf{c}' = [c_{n-n'}, c_{n-n'+1}, \ldots c_{n-n'-1}]$. Cyclic codes work on polynomials, i.e., instead of vector notation, the codeword is described as a polynomial

$$c(x) = c_0 + c_1 x + \ldots + c_{n-1}x^{n-1} \tag{7.4}$$

with binary coefficients $c_i$ and a degree of $n-1$ or less. The *generator polynomial* $g(x)$ of a linear $(n, k, t)$ cyclic code is the unique code polynomial of degree $n - k$

$$g(x) = 1 + g_1 x + \ldots + g_{n-k}x^{n-k} \tag{7.5}$$

and every other code polynomial can be expressed as $v(x) = u(x)g(x)$, from a message polynomial $u(x)$ of degree $k$ or less. The generator matrix $\mathbf{G}$ and parity check matrix $\mathbf{H}$ can be constructed from the generator polynomial $g(x)$ [45]. However, for larger codes the storage of the matrices becomes increasingly expensive and encoding and decoding based on polynomial operations is usually preferred for cyclic codes.

**Encoding of Binary BCH Codes** For any positive integers $m \geq 3$ and $t < 2^{m-1}$, there exists a binary $(n, k, t)$ BCH code with block length $n = 2^m - 1$, that is able to correct up to $t$ errors in the received codeword $\mathbf{c}'$ such that the corrected message block is $\hat{\mathbf{u}} = \mathbf{u}$. The generator polynomial $g(x)$ of a $t$-error correcting binary BCH code is defined as the lowest degree polynomial with binary coefficients that has the consecutive powers $\alpha, \alpha^2, \ldots, \alpha^{2t}$ of the primitive element $\alpha \in GF(2^m)$ as its roots, i.e., $g(\alpha^i) = 0$ for $1 \leq i \leq 2t$. With the minimal polynomial $\phi_i(x)$ of $\alpha^i$, the generator polynomial is

$$g(x) = \text{LCM}\{\phi_1(x), \phi_2(x), \ldots \phi_{2t}(x)\}, \tag{7.6}$$

where LCM is the Least Common Multiple (LCM). As even powers of the primitive element $\alpha$ share the same minimal polynomial, i.e., $\phi_{2i}(x) = \phi_i(x)$ for $\alpha^i$ and $\alpha^{2i}$, Eq. (7.6) can be reduced to

$$g(x) = \text{LCM}\{\phi_1(x), \phi_3(x), \ldots \phi_{2t-1}(x)\}. \tag{7.7}$$

Note that the degree of the minimal polynomials $\phi_i(x)$ is at most $m$, i.e., $\deg g(x) = n - k \leq mt$ with equality if and only if all minimal polynomials have degree $m$.

For systematic BCH codes that are usually employed in hardware implementations, the message part of length $k$ is separated from the redundant checking part of length $n - k$. The encoding of a message polynomial $u(x)$ is achieved by using the remainder of the polynomial division $r(x) = \left(u(x) \cdot x^{n-k}\right) \bmod g(x)$ as the parity part of the message, i.e., the systematic codeword polynomial is composed of the remainder polynomial $r(x)$ and the shifted message polynomial $u(x)$ as $c(x) = r(x) + u(x) \cdot x^{n-k}$. In hardware, the division to obtain the remainder $r(x)$ is efficiently implemented by a Galois LFSR with taps according to the coefficients of the generator polynomial $g(x)$. The message is shifted bit-wise into the LFSR, and after $n$ cycles the remainder is contained in the registers.

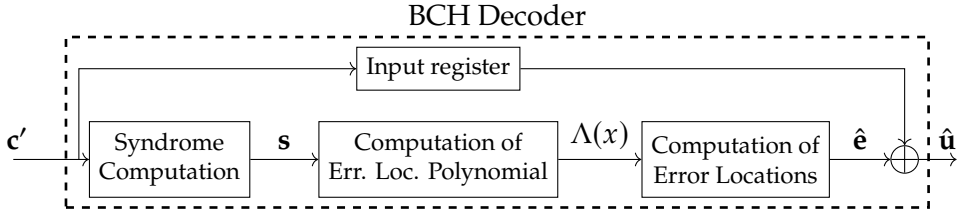



**Figure 7.1** General structure of a BCH decoder.

**Decoding of Binary BCH Codes** Fig. 7.1 depicts the general structure of a binary BCH decoder that consists of three steps: computing the syndrome $\mathbf{s}$, finding the error locator polynomial $\Lambda(x)$, and estimating the error positions $\hat{\mathbf{e}}$. Finally, the estimated error positions are used to retrieve the corrected message $\hat{\mathbf{u}}$ by adding them to the noisy codeword $\mathbf{c}'$ that is stored in the input register.

The syndrome computation step derives the syndrome $\mathbf{s} = [s_1, s_2, \ldots, s_{2t}]^T$ consisting of $2t$ components by evaluating the received codeword $c'(x)$ at the powers of the primitive element

$$s_i = c'(\alpha^i) \tag{7.8}$$

for $1 \leq i \leq 2t$ and with $s_i \in GF(2^m)$. The received polynomial, which consists of the original codeword polynomial $c(x)$ and the error polynomial $e(x)$, can be represented as

$$c'(x) = c(x) + e(x) = a_i(x)\phi_i(x) + b_i(x), \tag{7.9}$$

i.e., as a multiple $a_i(x)$ of the minimal polynomial $\phi_i(x)$ and a remainder

$$b_i(x) = \frac{c'(x)}{\phi_i(x)} \tag{7.10}$$

with $\deg b_i(x) < \deg \phi_i(x)$. Combining Eqs. (7.7) to (7.10) yields

$$s_i = c'(\alpha^i) = e(\alpha^i) = b_i(\alpha^i) = \left.\frac{c'(x)}{\phi_i(x)}\right|_{x=\alpha^i}, \tag{7.11}$$

as $\phi_i(\alpha^i) = 0$ and $c(\alpha^i) = 0$. In other words, the syndrome components $s_i$ can be obtained by dividing the received codeword polynomial by the respective minimal polynomial $\phi_i(x)$ and evaluating the remainder at the corresponding power $\alpha^i$ of the primitive element instead of evaluating the entire codeword polynomial. This allows for an efficient implementation in hardware as the polynomial division can be realized as an LFSR. As the primitive polynomials of odd and even powers are identical, the even syndrome components can be derived as $s_{2i} = s_i^2$, which further reduces the computational effort.

If the syndromes are $\mathbf{s} = \mathbf{0}$, no errors are detected and decoding is trivial. Otherwise, the error locator polynomial $\Lambda(x)$ is calculated from the syndrome components in Eq. (7.11). The error locator polynomial has the inverse of the error locations as its roots and can be retrieved by different algorithms. Usually, the Berlekamp-Massey Algorithm (BMA) is used, as it is an iterative and efficient method for solving the set of equations defined by the syndrome components. From the error locator polynomial, the error positions are determined by finding the roots of $\Lambda(x)$ using the Chien search that searches the roots by trial and error [45]. As in the following, the main focus is on the syndrome calculation, the BMA and the Chien search are not further detailed.

## 7.2 Hardware Implementation of BCH Codes

For a systematic SCA evaluation of BCH codes, implementations with different code parameters are required. Previous SCA attacks [49, 68] target a single code from a framework optimized for

communication applications, i.e., with a low latency of the design [105]. While SCA attacks on the auto-generated code are feasible, it is not optimized regarding readability. This makes it difficult to analyze more complex attack vectors than the input register of the BCH code. As in the PUF context area is more important than latency, in Section 7.2.1 a framework optimized for area-efficient implementations [103] is introduced. Furthermore, the generated code is structured into modules that allow for analyzing different steps of the BCH decoding separately. Section 7.2.2 provides the implementation of the syndrome computation that is used to refine the power model in Section 7.4.1.

### 7.2.1 BCH Code Generation Framework

The framework for generating BCH codes [103] is denoted as *Code Generation Framework (CGF)* in the following. The general structure of the generated code follows Fig. 7.1, where the decoding steps are carried out sequentially, i.e., the BMA and the Chien search start after the syndrome computation is finished. The CGF allows for selecting the code length $n$, input data size $k$ and the correction capability $t$ of the desired code. Furthermore, a number of test vectors can be selected that is generated along with the code implementation. The output of the CGF are a VHDL implementation of the systematic $(n, k, t)$ BCH code, a text file with the test vectors, and a corresponding test bench.

In the following, the design choices for the BMA and the Chien search are briefly outlined. The implementation of the syndrome computation is explained in more detail in Section 7.2.2 as it is used for the improved power model in Section 7.4.1.

The *BMA* is based on the Further optimized inversion-free Berlekamp-Massey (FiBM) [106] implementation, an inversion-less BMA implementation. By selecting a folding factor $K$, the parallelism of the implementation is determined, and a trade-off between latency and area can be chosen. Depending on the folding factor the number of identical Processing Elements (PEs) [107] is determined. For a fully-folded FiBM implementation, only a single PE is required, reducing the overall footprint [103]. Additionally, two shift registers of size $t + \lceil \frac{t}{2} \rceil$ and a control logic are required.

The *Chien search* substitutes the powers of the primitive element $\alpha^i$ in the error locator polynomial $\Lambda(x)$ determined by the BMA. If the result is zero, the power corresponds to a root of the error locator polynomial and the corresponding bit in the codeword is corrected. The hardware implementation from the CGF uses constant multipliers and adders, where the number of components is determined by the correction capability $t$ [103]. Note that the *message buffer* stores only the $k$ message bits and not the entire $n$-bit codeword, i.e., the Chien search only corrects the errors in the message part. As the parity part of the codeword does not contain information, this approach reduces the latency and saves hardware resources as less registers are needed in the message buffer.

### 7.2.2 Syndrome Computation: Division and Substitution

Hardware implementations allow for using LFSRs to calculate the polynomial division in Eq. (7.11) for the different syndrome components: The LFSR registers are initialized with zeros. The message is sequentially shifted into the LFSRs one bit per clock cycle such that the syndrome is obtained after $n$ clock cycles. The minimal polynomials $\phi_i(x)$, which specify the generator polynomial of the code according to Eq. (7.7), define the positions of the feedback taps and the number of LFSR registers. For a BCH code from the Galois field $GF(2^m)$, the minimal polynomials are of degree $\deg \phi_i(x) \leq m$, i.e., each LFSR consists of at most $m$ registers. The number of parallel LFSRs depends on the number of different minimal polynomials $N_\phi$, which is at most $t$. Note that the LFSRs only divide the received codeword polynomial $c'(x)$ by the corresponding minimal polynomials $\phi_i(x)$. In order to evaluate the result at $\alpha^i$ from the final state of the registers, an additional combinatorial logic circuit is required. As even powers of $\alpha$ share the same minimal polynomial, e.g., $\phi_1(x) = \phi_2(x) = \phi_4(x) = \ldots$, from a single LFSR structure multiple syndrome components are derived by different logic circuits.[1]

[1]Note that there is also the possibility of a shift register structure that evaluates $c'(x)$ directly at $\alpha^i$ [45]. For minimal polynomials that are used for single syndrome component only, the structure saves the additional combinatorial logic. In the following, this structure is not considered to simplify the analysis. As the BMA is by far the most resource demanding

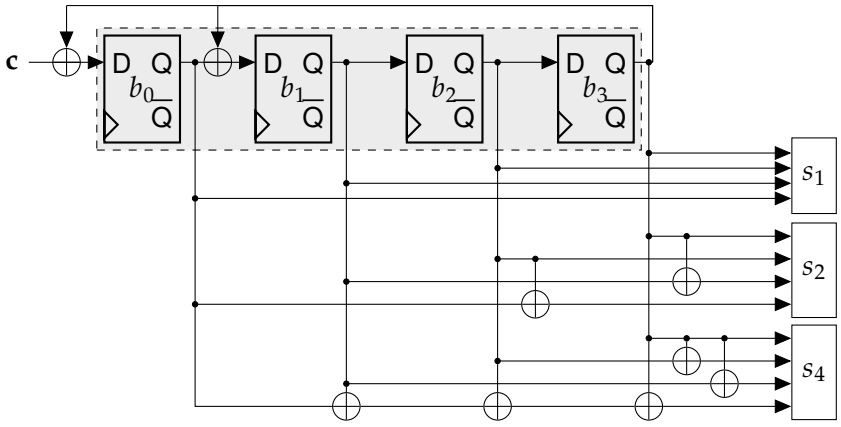



**Figure 7.2** Circuit to calculate the syndrome components $s_1$, $s_2$ and $s_4$ of a $(15, 7, 2)$ BCH code. Adapted from [45].

Fig. 7.2 depicts an example LFSR structure for the syndrome components $s_1$, $s_2$ and $s_4$ of a $(15, 7, 2)$ BCH code that is constructed from $GF(2^4)$. Consequently, there are $m = 4$ registers and the feedback taps of the Galois LFSR are defined by the minimal polynomial $\phi_1(x) = \phi_2(x) = \phi_4(x) = x^4 + x^3 + 1$. Finally, the register states are logically combined to form the syndrome components. The required logic connections are determined at design time from Eq. (7.11).

## 7.3 Differential Power Analysis of BCH Codes for PUF-based Key Generation

This section introduces the general concept of correlation-based DPA attacks in Section 7.3.1. Subsequently, the power model and helper data manipulation strategy of existing DPA attacks on BCH codes used for PUF-based key generation are outlined in Section 7.3.2.

### 7.3.1 Differential Power Analysis (DPA)

DPA attacks exploit power consumption differences of implementations for different data that is processed. By comparing the recorded power consumption to a model of the power consumption for different data, the most likely processed data is determined [51, 108]. If the comparison of hypothetical power consumption and measured power traces is done by Pearson's correlation coefficient $\rho$, the attack is also denoted as Correlation Power Analysis (CPA) [109]. The correlation coefficient is dimensionless and limited between $-1 \leq \rho \leq 1$, where independent variables result in a correlation coefficient of zero.

First, a suitable intermediate value of the attacked algorithm is chosen. Given $N_t$ measurements with different input data $d_i$ and $N_K$ possible values for the targeted intermediate value $k_j$, a $N_t \times N_K$ matrix $\mathbf{V}$ is generated. Each entry $v_{i,j}$ in $\mathbf{V}$ is derived from a model function $f(d_i, k_j)$ that determines the internal state of the implementation from the known input data $d_i$ and a possible candidate $k_j$ for the intermediate. A power model maps the state of the implementation in $\mathbf{V}$ to hypothetical power consumption values $\mathbf{H}$. The most common power model for hardware implementations is the *Hamming distance model* that assumes a power contribution for transitions $0 \rightarrow 1$ and $1 \rightarrow 0$ and ignores static power consumption, i.e., $0 \rightarrow 0$ and $1 \rightarrow 1$ do not contribute. The power model is

part of a BCH hardware implementation, the minimal saving of the combinatorial logic does not impact the overall hardware footprint significantly.

applied to all considered signals of the implementation, and the overall power model consists of the sum of all Hamming distances

$$P_{HD} = \sum_i \mathrm{HD}(v_i, v_{i,prev}), \tag{7.12}$$

where $v_i$ is the current value and $v_{i,prev}$ is the previous value of the signal or register.

The measurements of the implementation are stored in a matrix $\mathbf{T}$ of size $N_t \times N_s$, where the traces of $N_s$ samples are organized in the rows of $\mathbf{T}$. The correlation coefficient for the hypothetical power consumption $\mathbf{H}$ and the measured traces $\mathbf{T}$ is calculated for each candidate $i$ and sample point $j$: The columns $\mathbf{h_i}$ and $\mathbf{t_j}$ are compared using the correlation coefficient

$$\rho_{i,j} = \frac{\sum_{d=1}^{N_t} \left(t_{d,j} - \bar{t}_j\right)\left(h_{d,i} - \bar{h}_i\right)}{\sqrt{\sum_{d=1}^{N_t} \left(t_{d,j} - \bar{t}_j\right)^2}\sqrt{\sum_{d=1}^{N_t} \left(h_{d,i} - \bar{h}_i\right)^2}}, \tag{7.13}$$

where $\bar{h}_i$ and $\bar{t}_j$ denote the mean values of $\mathbf{h_i}$ and $\mathbf{t_j}$ taken over the different input data values. The values $\rho_{i,j}$ are combined in the correlation matrix $\mathbf{P}$, where the row index $i$ corresponds to the value of the candidate and the column index $j$ corresponds to the time sample of the measured trace.

The maximum correlation coefficients for the possible candidates are obtained by taking the maximum over the samples of $\mathbf{P}$. Usually, all samples $\mathcal{S} \in \{0, \ldots, N_s - 1\}$ are included in the evaluation. However, if the point in time within the measurements at which the operation of interest is executed is known, only a subset $\mathcal{S} \in \{s_{min}, \ldots, s_{max}\}$ of the time samples can be included. Limiting the evaluated time span reduces the computational effort. The candidate with the maximum correlation coefficient

$$i_{cand} = \arg\max_{0 \le i \le N_K - 1} \max_{j \in \mathcal{S}} \mathbf{P}, \tag{7.14}$$

determines the guess of the attack.

### 7.3.2 DPA of BCH Codes

Hardware implementations of concatenated BCH and repetitions codes have been shown to be vulnerable to DPA attacks, where the input register of the BCH code is targeted [49, 68]. Existing attacks consider the concatenation of an ($n_{rep} = 7$, $k_{rep} = 1$, $t_{rep} = 3$)-repetition code with an ($n = 127$, $k = 64$, $t = 10$) BCH code, where two decodings generate a 128-bit key [49, 68]. Consequently, the number of PUF response and helper data bits is $2 \times 127 \times 7$, where the input bits of the BCH decoder are generated from the decoding of the repetition code bits as

$$\begin{aligned} c_i &= \mathrm{Decode}_{\mathrm{Rep}}(\mathbf{r'_{rep,i}} \oplus \mathbf{w_{rep,i}}) && (7.15)\\ &= \mathrm{Decode}_{\mathrm{Rep}}([r'_{i \cdot n_{rep}}, \ldots r'_{(i+1)\cdot n_{rep}-1}] \oplus [w_{i \cdot n_{rep}}, \ldots w_{(i+1)\cdot n_{rep}-1}]) && (7.16)\\ &= \mathrm{Decode}_{\mathrm{Rep}}(\mathbf{c_{rep,i}}) = \begin{cases} 0 & \text{if } \mathrm{HW}(\mathbf{c_{rep,i}}) < \lceil n_{rep}/2 \rceil \\ 1 & \text{if } \mathrm{HW}(\mathbf{c_{rep,i}}) \ge \lceil n_{rep}/2 \rceil \end{cases}, && (7.17) \end{aligned}$$

i.e., always $n_{rep} = 7$ bits of the PUF response and the helper data derive a bit that is processed by the BCH decoder. By changing the values of helper data bits, single bits of $\mathbf{c_{rep,i}}$ can be modified and a DPA attack can be tailored that attacks the value of the BCH code bit $c_i$ in the input register, respectively the processing of the results of the repetition code.

**Requirement and Assumptions for the Attack** There are some requirements and assumptions for the DPA attack on the BCH decoder:

- While the codeword $\mathbf{c}$ processed by the decoder is the attack target, due to $\mathbf{c} = \mathbf{r} \oplus \mathbf{w}$, recovering the PUF response $\mathbf{r}$ is equivalent as the codeword can be obtained by combining $\mathbf{r}$ with the public helper data $\mathbf{w}$.

- The same PUF response is assumed for all SCA measurements that are acquired, which is generally the case for PUF-based key generation. Note that actually the noisy PUF response $\mathbf{r}' = \mathbf{r} \oplus \mathbf{e}$ is processed in each trace, i.e., the targeted secret is subject to an error $\mathbf{e}$. However, assuming that the PUF is stable under nominal conditions and that the attacker keeps the environment at the nominal conditions, bit errors are neglected in the following.[2]
- In order to enable DPA attacks, the helper data has to be publicly known and its manipulation must be possible. Note that for the Fuzzy Commitment Scheme (FCS) [4] and the related Code-Offset Fuzzy Extractor (COFE) [12] the manipulation of the helper data affects directly the processed codeword. For other HDAs from Section 2.1.2, such as syndrome and parity constructions, the influence of the helper data manipulation on the processed data may be different, i.e., a modification of the hypothesis generation is needed.
- Errors $\mathbf{e_a}$ in the recovered codeword $\hat{\mathbf{c}} = \mathbf{c} \oplus \mathbf{e_a}$ compared to the correct codeword $\mathbf{c}$ can be corrected as long as $\mathrm{HW}(\mathbf{e_a}) \leq t$, i.e., if their number is smaller than the correction capability of the code under attack.
- The registers of the BCH decoder are assumed to be initialized with all zeros before the first bit is processed, i.e., the previous values of all registers for the first bit are $v_{-1} = 0$.

**Input Register Power Model** Existing attacks use the input register of the BCH decoder as intermediate value, but target the processing of several bits of a repetition code to obtain the register value [49, 68]. The attack on bit $c_i$ requires the preceding value of the input register $v_{i-1} = \hat{c}_{i-1}$ from the previous step of the attack to calculate the Hamming distance power model as

$$P_{input}\left(c_i, v_{i-1}\right) = \mathrm{HD}\left(c_i, v_{i-1}\right). \tag{7.18}$$

Note that the switching of codeword bits $c_0, \ldots, c_{i-2}$ does not depend on $c_i$ and is therefore omitted in the power model. As the codeword is sequentially processed by the decoder, the bits of the codeword are attacked sequentially as well.

**Helper Data Manipulation** The manipulation of helper data $\mathbf{w_{man}}$ affects directly bits of the codeword

$$\mathbf{c}' = \mathbf{r}' \oplus \mathbf{w} \oplus \mathbf{w_{man}}, \tag{7.19}$$

that is processed by the BCH decoder, where $\mathbf{w_{man}}$ is the vector with the manipulated helper data. Each bit can be manipulated by $x \in \{0, 1\}$, i.e., for each helper data bit $w_i \oplus x$ either its public value is used ($x = 0$) or it is flipped ($x = 1$). The DPA attack on the BCH decoder takes the manipulated helper data $\mathbf{w} \oplus \mathbf{w_{man,j}}$ as known input data for the $j^{\text{th}}$ trace: The helper data $\mathbf{w}$ is public and the bit manipulations $\mathbf{w_{man,j}}$ are controlled by the attacker.

The original attack iterates over all $2^{n_{rep}}$ values of the input of the repetition code to find the most probable value of the $n_{rep}$-bit PUF response chunk corresponding to the respective helper data [49]. Given the $n_{rep}$-bit candidate $\hat{\mathbf{r}}_{\mathbf{rep,i}}$, the $i^{\text{th}}$ bit $c_i$ of the BCH code is determined by decoding $\hat{\mathbf{c}}_{\mathbf{rep,i}} = \hat{\mathbf{r}}_{\mathbf{rep,i}} \oplus \mathbf{w}_{\mathbf{rep,i}}$ according to Eq. (7.17), where $\mathbf{w}_{\mathbf{rep,i}}$ is the value of the public helper data.

The manipulation of the repetition code helper data $\mathbf{w}_{\mathbf{man,rep,j}}$ loops over the $2^{n_{rep}}$ different values. As a result, the corresponding BCH codeword bit is altered according to the result of the decoding $\mathrm{RepDec}\left(\mathbf{w}_{\mathbf{man,rep,j}}\right)$ of the respective manipulation. After $2^{n_{rep}}$ manipulations, the next $n_{rep}$-bit chunk

[2] In case bit errors $\mathbf{e}$ occur, the respective trace will not match the power model anymore, which can be seen as noise for the DPA. This would increase the number of required traces, but does not hinder the DPA generally.

of helper data, corresponding to the bit $c_{i+1}$ of the BCH code, is manipulated. Consequently, the manipulation of the BCH codeword for the $j^{\text{th}}$ trace is

$$\mathbf{w_{man,j}} = \begin{cases} [x,0,0,\ldots,0,0,0] & \text{if } 0 \leq j \leq 2^{n_{rep}}-1 \\ [0,x,0,\ldots,0,0,0] & \text{if } 2^{n_{rep}} \leq j \leq 2\cdot 2^{n_{rep}}-1 \\ \ldots & \ldots \\ [0,0,0,\ldots,0,0,x] & \text{if } (n-1)\cdot 2^{n_{rep}} \leq j \leq n\cdot 2^{n_{rep}}-1 \end{cases}, \tag{7.20}$$

where $x$ is the manipulated value according to

$$x = \begin{cases} 0 & \text{if HW}\left((j \bmod 2^{n_{rep}})_2\right) < \lceil n_{rep}/2 \rceil \\ 1 & \text{if HW}\left((j \bmod 2^{n_{rep}})_2\right) \geq \lceil n_{rep}/2 \rceil \end{cases} = \text{RepDec}\left((j)_2 \bmod 2^{n_{rep}}\right) \tag{7.21}$$

and $(\cdot)_2$ is the binary representation of the trace index. In other words, the manipulation pattern mimics the decoding of the repetition code from Eq. (7.17).

Taking into account that not the value of the $n_{rep}$-bit PUF response chunk but the value of the BCH codeword bit is of interest, the possible values for the $n_{rep}$-bit chunk can be reduced to one bit. The original attack has been modified, such that the correlations from other candidates are leveraged to increase the robustness of the attack by matching on the characteristic correlation pattern of all candidates [68]. Yet, $2^{n_{rep}}$ manipulated helper data bits are required for each bit of the BCH decoder input [49, 68]. Conduction manipulations for every second bit in the same measurement minimizes the number of measurements [68], and the manipulation pattern is changed to

$$\mathbf{w_{man,j}} = \begin{cases} [x,0,x,\ldots,x,0,x] & \text{if } 0 \leq j \leq 2^{n_{rep}}-1 \\ [0,x,0,\ldots,0,x,0] & \text{if } 2^{n_{rep}} \leq j \leq 2\cdot 2^{n_{rep}}-1 \end{cases}. \tag{7.22}$$

Note that with the manipulation strategy from Eq. (7.22) more than $t$ bits in $\mathbf{c}$ are changed, i.e., the decoded key is $\tilde{\mathbf{k}} \neq \mathbf{k}$ and the manipulation can be detected by the device.

Using the input register power model and the manipulation of the helper data of the repetition decoder is limited to concatenated code constructions. Therefore, Section 7.4 improves the existing attacks by a more sophisticated power model and a modified helper data manipulation strategy, making them applicable to stand-alone BCH codes. Additionally, in order to overcome the requirement of helper data manipulation altogether, a horizontal SCA attack is proposed.

## 7.4 Extended SCA Attacks on BCH Codes

This section extends the existing DPA attack on BCH codes from Section 7.3.2 in two directions. First, Section 7.4.1 improves the power model by including the syndrome computation in addition to the input buffer, and adapts the helper data manipulation pattern. The resulting attack does not rely on concatenated codes, but can be applied to stand-alone BCH codes. Second, Section 7.4.2 introduces a correlation-based Horizontal Side-Channel Analysis (HSCA) attack that combines different parts of a single trace. Consequently, helper data manipulation is not needed, and the attack is more powerful than DPA attacks. The attacks from Sections 7.3.2, 7.4.1 and 7.4.2 are compared in Section 7.4.3. Finally, Section 7.4.4 introduces the possibility to correct additional errors in the attack by exploiting the linear and cyclic properties of BCH codes.

### 7.4.1 Improved DPA Attack

The DPA attack from Section 7.3.2 requires the concatenation of the targeted BCH code with a repetition code in order to enable the proposed helper data manipulation. Furthermore, only the input register is included into the power hypothesis, such that the power from the syndrome computation is not covered by the model. In order to overcome these shortcomings, the power model is extended to include the LFSR state of the syndrome computation. Furthermore, the helper data manipulation is modified such that no concatenated repetition code is required and the decoder cannot detect the manipulation.

**Syndrome LFSR Power Model** The power model from Eq. (7.18) is very limited given that the syndrome computation from Section 7.2.2 is carried out in parallel. Therefore, the improved DPA attack uses a more complex power model based on the LFSRs of the syndrome computation. In the following, the update of the LFSR registers from state $\mathbf{v}_{i-1}$ to state $\mathbf{v}_i$ by shifting bit $c_i$ into the LFSR are modeled as a matrix multiplication

$$\mathbf{v}_i = \mathbf{A} \begin{bmatrix} c_i \\ \mathbf{v}_{i-1} \end{bmatrix}, \tag{7.23}$$

where $\mathbf{A}$ is the *state transition matrix* for the LFSR defined by a minimal polynomial $\phi(x)$ from Eq. (7.7).[3] Note that the registers are initialized with zeros before the first bit $c_0$ is added, i.e., $\mathbf{v}_{-1} = \mathbf{0}$.

The minimal polynomials used to calculate the syndromes according to Eq. (7.11) are of the form $\phi(x) = a_M x^M + \ldots + a_1 x + 1$ with degree $M := \deg \phi(x)$. The feedback vector $\mathbf{a}$ is derived from the coefficients of $\phi(x)$ as $\mathbf{a} = [1, a_1, \ldots, a_{M-1}]^T$. The $M \times (M+1)$ state transition matrix is

$$\mathbf{A} = \left[\mathbf{I}|\mathbf{a}\right] = \begin{bmatrix} 1 & 0 & \ldots & 1 \\ 0 & 1 & \ldots & a_1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & a_{M-1} \end{bmatrix}, \tag{7.24}$$

i.e., the $M \times M$ identity matrix $\mathbf{I}$ represents the shift to the next register and the feedback vector $\mathbf{a}$ adds the feedback of the Galois LFSR from the last register. Using the Hamming distance model and following Eq. (7.23), the power consumption of a single LFSR is

$$P_{LFSR} = \mathrm{HD}\left(\mathbf{v}_i, \mathbf{v}_{i-1}\right) \tag{7.25}$$

and the overall power consumption from the $N_\phi \leq t$ different LFSRs results in

$$P_{LFSRs} = \sum_{p=0}^{N_\phi - 1} P_{LFSR,p}. \tag{7.26}$$

Note that the improved power model requires the correct values of preceding bits $[c_0, \ldots, c_{i-1}]$ in order to iteratively calculate the previous state $\mathbf{v}_{i-1}$ from Eq. (7.23). As the initial values $\mathbf{v}_{-1} = \mathbf{0}$ are known, the requirement is fulfilled for the first bit. For all subsequent bits, the values are derived from the previously attacked bits $[\hat{c}_0, \ldots, \hat{c}_{i-1}]$.

**Improved Helper Data Manipulation** DPA attacks could be hampered by detecting erroneous decodings and locking the device, e.g., by allowing only a limited number of failed decodings. Consequently, the approach of parallel manipulation from Eq. (7.22) would be detected. It could be modified to manipulate at most $t$ bits at once, i.e., the number of manipulated bits has to be less than the error correcting capability of the BCH decoder. This would increase the number of traces by a factor of $\left\lceil \frac{n}{t} \right\rceil$. Furthermore, the helper data manipulation approaches from Section 7.3.2 rely on the existence of a concatenated repetition code, and the number of traces with a different manipulation pattern is fixed to $N_t = 2^{n_{rep}}$ per attacked bit, i.e., extracting more information for the DPA attack is not possible with the approach.[4] In combination with the simple input register power model, attacks on stand-alone BCH codes are not possible.

The improved DPA attack uses a different helper data manipulation compared to Eqs. (7.20) and (7.22). Instead of a systematic manipulation pattern, $t$ bits are randomly manipulated for each measurement

$$\mathbf{w_{man,j}} = [x_0, x_1, \ldots, x_{n-1}], \; x_i \xleftarrow{\$} [0,1], \text{ s.t. } \mathrm{HW}\left(\mathbf{w_{man,j}}\right) = t, \tag{7.27}$$

[3] Note that in order to avoid confusion with the codeword bit index $i$, the index of $\phi$ is omitted in this paragraph.

[4] Note that repeated measurements can be used to increase the SNR [68].

which ensures that the decoding of the BCH codeword does not fail. In combination with the syndrome LFSR power model, the improved manipulation can be applied to BCH codes independent of their concatenation to other codes. The number of different manipulation patterns according to Eq. (7.27) is upper limited by $\binom{n}{t}$, i.e., by the possibilities to distribute the $t$ manipulated bits among the $n$ bits of the codeword. In contrast to the attack from Section 7.3.2, the number of traces $N_t$ of the DPA attack can be chosen independently, i.e., there is no minimal number of traces that is required.

### 7.4.2 Horizontal Side-Channel Analysis

Horizontal Side-Channel Analysis (HSCA) is a powerful class of SCA attacks that combines the power consumption of different parts of a trace [110, 111].[5] Instead of aggregating information from different measurements with varying input data, as for DPA attacks, HSCA is performed along the time domain by combining different points in time of a single trace. The prerequisite for HSCA is that the secret under attack is processed in different parts of the algorithm, and consequently appears in different parts of the power trace.

**Requirements** For the scenario of attacks on BCH codes used for PUF-based key generation, HSCA provides several improvements compared to the DPA attacks from Section 7.4.1.

- As HSCA operates on a single trace, the manipulation of helper data to generate differential measurements is not required. Consequently, even if the write access to helper data is hampered, attacks on a single trace with known helper data are still possible.
- While for vertical attacks noisy PUF measurements, i.e., where $\mathbf{e} \neq \mathbf{0}$, can constitute a problem as the secret is not constant across measurements, HSCA is not influenced by bit errors in the PUF response. As it operates on a single trace, it recovers the noisy codeword, which can be corrected to the secret key as long as the number of errors is within the correction capability $t$.

**Correlation-Based Horizontal Side-Channel Attack** The HSCA on the BCH decoder compares the power consumption of different points in time with the hypothetical power consumption for the possible input sequences. Similar to DPA in Section 7.4.1, Pearson's correlation coefficient is leveraged to evaluate the similarity of the observed power consumption and the hypothetical values [104].

As the bits are serially shifted into the BCH decoder, a new codeword bit is added in each clock cycle. The correlation-based HSCA attack targets sequences of $m$ bits at once by modeling the expected power consumption over $m$ clock cycles with the LFSR power model from Section 7.4.1. There are $2^m$ different sequences that have to be considered as candidates, i.e., the hypothesis matrix $\mathbf{H}$ is of size $2^m \times m$. The number of clock cycles $m$ that is modelled for each candidate is denoted as *hypothesis length* in the following. For a longer hypothesis length, more information from different points in time is included into the attack. However, the hypothesis matrix $\mathbf{H}$ grows exponentially in $m$, limiting the hypothesis length by the available computing resources.

From the power trace $\mathbf{t}$ with $N_s$ samples, a sample for each clock cycle is selected resulting in a trace segment $\mathbf{t}^\star$ of $m$ samples.[6] Similar to Eq. (7.13), the correlation coefficient between the rows $\mathbf{h_i}$ of the hypothetical power consumption $\mathbf{H}$ and the trace segment $\mathbf{t}^\star$ is calculated for each candidate $i$:

$$\rho_i = \frac{\sum_{d=1}^{m} \left(t_d^\star - \overline{t^\star}\right)\left(h_{d,i} - \bar{h}_i\right)}{\sqrt{\sum_{d=1}^{m} \left(t_d^\star - \overline{t^\star}\right)^2} \sqrt{\sum_{d=1}^{m} \left(h_{d,i} - \bar{h}_i\right)^2}}, \tag{7.28}$$

[5] As the time domain is usually plotted on the x-axis, *horizontal* refers to the fact that HSCA operates on shares from different points in time. In contrast, *vertical* attacks use the same point in time from multiple traces.

[6] For simulated results, there are no alignment issues due to jitter. Note that for practical measurements it is also possible to select more than one sample per clock cycle at the cost of additional computations: the samples have to the mutually combined, i.e., for $N'$ samples per clock cycle $N'^m$ combinations exits for which a separate correlation according to Eq. (7.28) has to be calculated.

**Table 7.1** Comparison of SCA attacks on BCH codes used for PUF-based key generation.

| | Power Model | Minimum # Traces | # Manipulations | Detectable | Repetition Code |
|---|---|---|---|---|---|
| Original Attack [58] | Input register | $n \times 2^{n_{rep}}$ | $n \times 2^{n_{rep}}$ | no | yes |
| Modified Attack [68] | Input register | $2 \times 2^{n_{rep}}$ | $n \times 2^{n_{rep}}$ | yes | yes |
| | Input register | $2 \times 2^{n_{rep}} \times \left\lceil \frac{n}{t} \right\rceil$ | $n \times 2^{n_{rep}}$ | no | yes |
| Improved DPA Attack | Syndrome LFSRs | $N_t$ | $N_t \times t$ | no | no |
| HSCA Attack | Syndrome LFSRs | 1 | – | no | no |

where $\bar{h}_i$ and $\overline{t^\star}$ denote the mean values of $\mathbf{h_i}$ and $\mathbf{t^\star}$ taken over the entire segment. The values $\rho_i$ are combined in the correlation vector $\mathbf{P}$, where the index $i$ corresponds to the value of the candidate sequence. The index with the highest correlation

$$i_{cand} = \underset{0 \leq i \leq 2^m - 1}{\arg\max} \mathbf{P} \tag{7.29}$$

provides the sequence determined by the attack. The bits $[\hat{c}_0, \ldots, \hat{c}_{m-1}]$ are set according to the sequence with the highest correlation. In the next step, the internal state of the LFSRs is calculated from the bits retrieved from the attack. The hypotheses for the following $m$ bits are computed, and the bits $[\hat{c}_m, \ldots, \hat{c}_{2m-1}]$ are determined analogously. In order to determine the entire $n$-bit codeword, the correlation-based attack is conducted $\left\lceil \frac{n}{m} \right\rceil$ times. As outlined in Section 7.4.1, the LFSR-based power hypothesis of subsequent sequences depends on the previously recovered codeword bits. Thus, if an error in estimating one of the bits occurs, the model of the state $\mathbf{v}_{i-1}$ of the BCH decoder is erroneous and subsequent hypotheses do not match.

**Increasing the Robustness of the Attack** The robustness of the attack can be increased by using only the first bit of the retrieved candidate sequence from Eq. (7.29). Consequently, the hypotheses have an overlap of $m-1$ positions with the hypotheses of preceding bits: The first bit is attacked with a hypothesis length $m$, and the power consumption of bits 0 to $m-1$ is included in the hypothesis. From the determined candidate only the first bit is used, i.e., the codeword guess is only updated by $\hat{c}_0$ instead of $[\hat{c}_0, \ldots, \hat{c}_{m-1}]$. The assumed internal state of the LFSRs is updated by the determined candidate bit. In the next step, the bits 1 to $m$ are included in the hypothesis, and $\hat{c}_1$ is retrieved. The procedure is repeated for the first $n-m$ bits. For the last $m$ bits, the entire sequence is adopted for $[\hat{c}_{n-m}, \ldots, \hat{c}_{n-1}]$ to avoid using shorter hypothesis length for the last bits. This method comes at the cost of an increased computational effort as instead of $\left\lceil \frac{n}{m} \right\rceil$ correlation computations, $n-m+1$ correlation computations are necessary. However, it also increases the robustness of the attack, and is therefore used in the following.

### 7.4.3 Comparison of SCA Attacks

Table 7.1 compares the SCA attacks from Sections 7.3.2, 7.4.1 and 7.4.2. The first three rows summarize the existing attacks from Section 7.3.2, including the modified attack that avoids detection of the manipulation as outlined in Section 7.4.1. Due to the BCH input register power model, the attacks rely on the concatenation with a repetition code. This also limits the number of different manipulation patterns and therefore the available traces.

The extended attacks from this thesis, presented in Sections 7.4.1 and 7.4.2, employ the more complex syndrome LFSR power model, which removes the requirement of a concatenated repetition code. The improved DPA attack from Section 7.4.1 manipulates only $t$ bits per trace. Consequently, as long as there are no additional errors in the PUF response, the BCH decoder corrects to the correct key and the manipulation cannot be detected. The HSCA attack does not require any helper data manipulations, and is done on a single trace.

### 7.4.4 Attacking the Difference Codeword

Finally, a method is introduced that allows for correcting additional errors of the attack by exploiting the linear and cyclic properties of BCH codes. From Section 7.1, for linear codes the sum $\mathbf{c}' = \mathbf{c}_1 \oplus \mathbf{c}_2$ of two codewords $\mathbf{c}_1, \mathbf{c}_2$ results in a valid codeword. Furthermore, for cyclic codes a cyclic shift of the codeword results in a valid codeword. Consequently, for linear cyclic codes the vector $\mathbf{d}$ resulting from the difference of subsequent bits of a codeword $d_i = c_i \oplus c_{i+1}$ is also a codeword denoted as *difference codeword* in the following [78].

The recovered codeword $\hat{\mathbf{c}} = \mathbf{c} \oplus \mathbf{e_a}$ is the correct codeword $\mathbf{c}$ subject to errors at positions defined by the attack error $\mathbf{e_a}$. The errors can be corrected as long as their number $\mathrm{HW}(\mathbf{e_a})$ is smaller than the correction capability $t$ of the code under attack. Now, the difference codeword is

$$\hat{\mathbf{d}} = [c_0 \oplus e_{a,0} \oplus c_1 \oplus e_{a,1},\ c_1 \oplus e_{a,1} \oplus c_2 \oplus e_{a,2},\ \ldots,\ c_{n-1} \oplus e_{a,n-1} \oplus c_0 \oplus e_{a,0}] \tag{7.30}$$

$$= \mathbf{d} \oplus \mathbf{d_a}, \tag{7.31}$$

i.e., the sum of the valid codeword $\mathbf{d}$ and the difference error $\mathbf{d_a}$ of the attack error $\mathbf{e_a}$. For the attack, the use of $\hat{\mathbf{d}}$ can be beneficial if $\mathrm{HW}(\mathbf{d_a}) \leq t < \mathrm{HW}(\mathbf{e_a})$, i.e., if the difference codeword $\hat{\mathbf{d}}$ is decodable, while the codeword $\hat{\mathbf{c}}$ itself is not. The necessary condition is that the errors in $\mathbf{e_a}$ are located in blocks of neighboring bits, e.g., for $\mathbf{e_a} = [001111110]$ the resulting difference error would be $\mathbf{d_a} = [001000010]$. If the errors are scattered over the codeword, the difference codeword does not provide a benefit, e.g., $\mathbf{e_a} = [011010110]$ results in $\mathbf{d_a} = [101111010]$.

If $\mathrm{HW}(\mathbf{d_a}) \leq t$, decoding the difference codeword $\hat{\mathbf{d}}$ from Eq. (7.30) and re-encoding the result leads to an error-free codeword

$$\mathbf{d} = [d_0,\ d_1\ \ldots,\ d_{n-1}] = [c_0 \oplus c_1,\ c_1 \oplus c_2,\ \ldots,\ c_{n-1} \oplus c_0], \tag{7.32}$$

from which the codeword $\mathbf{c}$ can be derived. Using the relation $c_{i+1} = c_i \oplus d_i$, the codeword bits are mutually related with each other. By fixing one bit $c_i$ to either 0 or 1, there are two possibilities for $\mathbf{c}$.

## 7.5 Simulated Side-Channel Analysis Results

This section provides simulated SCA results for the improved DPA attack and the HSCA attack from Sections 7.4.1 and 7.4.2. In Section 7.5.1 the experimental setup with power trace simulation and noise simulation is introduced. The results for the improved DPA attack in Section 7.5.2 show that it is robust against noise. In contrast in Section 7.5.3, the horizontal attack requires a perfect match of the power traces and the model as well as low noise levels, i.e., it is subject to practical limitations.

### 7.5.1 Experimental Setup

In order to experimentally analyze the SCA attacks, power traces from a behavioral simulation are generated. By using simulated measurements, the attacks can be carried out under controlled noise conditions to verify their robustness.

#### Power Trace Simulation

The workflow for generating traces for the analysis is depicted in Fig. 7.3. It allows for generating simulated power traces for arbitrary binary, systematic BCH codes. For the evaluation, four different codes are considered, where the $(127, 64, 10)$ code and the $(255, 131, 18)$ code are selected according to Table 2.1. The $(63, 36, 5)$ code has a similar code rate $k/n$ as the first two codes, and the $(15, 5, 3)$ code is selected as an example for a small code with a lower code rate. Appendix D provides the minimal polynomials of the selected codes that define the LFSR structure for the power model.

First, the Code Generation Framework [103] from Section 7.2.1 generates a Register-Transfer Level (RTL) description of the BCH decoder, a set of test vectors, and a test bench. The CGF takes

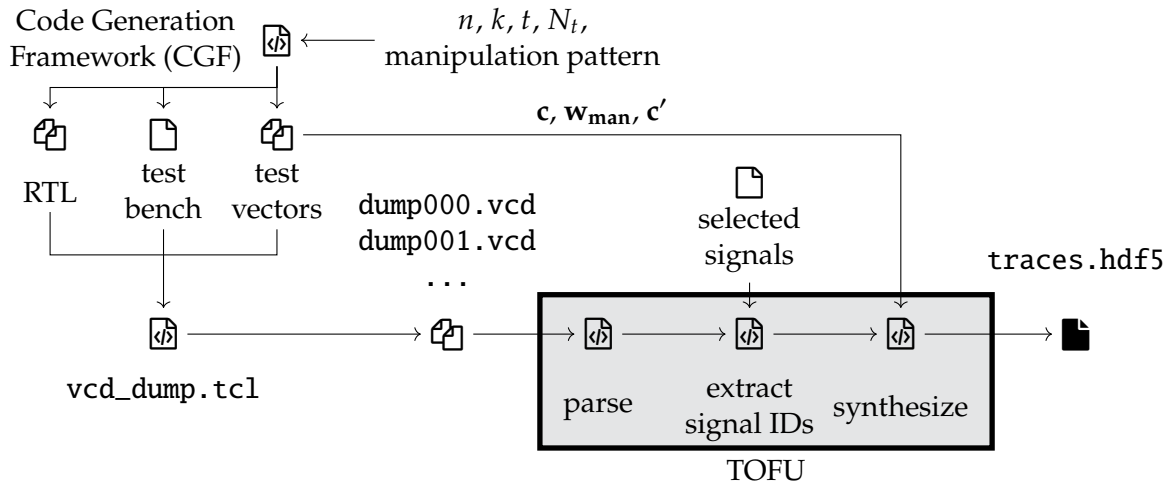



**Figure 7.3** Workflow for generation of simulated power traces.

as inputs the code parameters $(n, k, t)$ and the number of test vectors, i.e., the number of traces $N_t$ for simulation. Different helper data manipulations $\mathbf{w_{man}}$ can be added to the codeword for vertical attacks. The RTL description is a set of VHDL files of the generated BCH code. A corresponding test bench allows for verifying the functional behavior of the RTL code for different test vectors. The test vectors consist of the secret codeword $\mathbf{c}$, the helper data manipulation $\mathbf{w_{man}}$, and the resulting processed codeword $\mathbf{c}' = \mathbf{c} \oplus \mathbf{w_{man}}$. From the test bench and the generated test vectors, an additional Tool command language (Tcl) script allows for generating Value Change Dump (VCD) simulation traces from a behavioral simulation.

Second, the TOggle Foul-Up (TOFU) tool [112] transforms the VCD simulation traces into power traces based on a Hamming distance leakage model. The tool consists of three steps: first the VCD values are parsed. In a second step, the tool allows for selecting only a subset of signals from the entire implementation, such that additional switching noise can be reduced. To specify the desired entities, a list of signal literals can be passed to the signal extraction step. Finally, the power traces are synthesized from the selected signals, resulting in a matrix of traces $\mathbf{T}$ that is stored in an Hierarchical Data Format 5 (HDF5) file. In addition, the correct and noisy codeword, as well as the helper data manipulation pattern for each trace are stored in the HDF5 file.

#### Noise Simulation

The power consumption of CMOS circuits consists of a static and a dynamic part, where the dynamic power consumption caused by switching of cells is usually targeted by SCA. From an attackers perspective, the power consumption is composed of different parts as

$$P_{total} = P_{exp} + P_{sw.noise} + P_{el.noise} + P_{const}, \tag{7.33}$$

where $P_{exp}$ is the exploitable power consumption, $P_{sw.noise}$ the switching noise, $P_{el.noise}$ inherent electronic noise of the circuit and $P_{const}$ a constant offset, independent of operations and data, e.g., caused by leakage currents [51]. The *constant offset* follows $P_{const} \sim \mathcal{N}(\mu_{const.}, 0)$ and the *electronic noise* is modeled as $P_{el.noise} \sim \mathcal{N}(0, \sigma)$, where $\mu_{const.}$ is the mean and $\sigma$ is the standard deviation. The *exploitable* power consumption corresponds to all data and operation dependent components of a circuit that are modeled by the attacker. In contrast, the *switching noise* is the part of the power consumption that is data and operation dependent, but not captured by the model.

The traces generated from the CGF and TOFU in Section 7.5.1 are noise-free and do not consider electronic noise terms typically contained in SCA measurements, i.e., the standard deviation of the electronic noise is $\sigma = 0$. The noise-free power simulations allow only for investigating the influence of the switching noise, i.e., to show that the attack works in principle. However, the robustness under

electronic noise is of interest to determine the practical threat of an attack. Thus, a noise term is added to the noise-free simulations in order to evaluate the influence of noisy measurements on the attack success.

Electronic noise is approximated well by a normal distribution [51]. Assuming further that the noise of different signals is uncorrelated, i.e., statistically independent, the noise is described as

$$P_{el.noise}^{model} = \sum_i \mathcal{N}(0, \sigma_i) = \mathcal{N}\left(0, \sqrt{\sum_i \sigma_i^2}\right) = \mathcal{N}(0, \sigma)\,, \tag{7.34}$$

i.e., the noise model combines all noise sources by a single normal distribution with the standard deviation $\sigma$. The noise is sampled independently for each point in time and is added to the samples of the noise-free power trace.[7]

### 7.5.2 Improved DPA Attack

In this section, the improved DPA attack from Section 7.4.1 is investigated. For all codes, $N_t = 10000$ traces are simulated with a fixed codeword that is randomly chosen for each code. All signals from the BCH decoder are included in the simulated power trace, i.e., the input register, the syndrome calculation, BMA and the Chien search. In order to enable the DPA attack, helper data manipulation of different bits per trace is a requirement. The improved manipulation pattern from Section 7.4.1 that changes $t$ bits in each trace is applied. Nine different noise levels ranging from $\sigma = 1$ to $\sigma = 200$ are simulated with ten repetitions per noise level. The repetitions ensure that the influence of favorable or unfavorable noise additions is minimized by taking the average attack success over all repetitions.

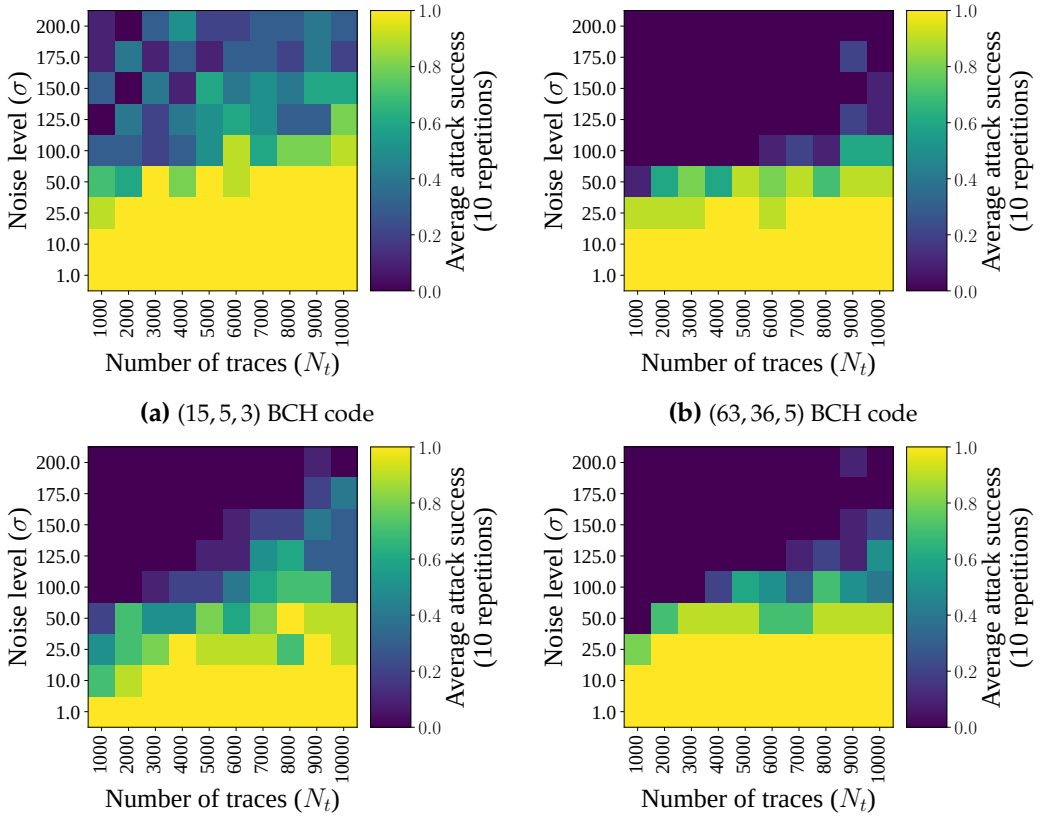
In order to determine the attack success, the number of errors $\mathbf{e_a} = \hat{\mathbf{c}} \oplus \mathbf{c}$ in the codeword $\hat{\mathbf{c}}$ retrieved by the attack compared to the correct codeword $\mathbf{c}$ is related to the correction capability $t$ of the code. If the number of errors introduced by the attack is less than the correction capability, the attacker can decode the secret key $\mathbf{k}$, and the attack is deemed successful. Following the difference codeword approach from Section 7.4.4, additionally the same comparison is done for the difference codeword $\hat{\mathbf{d}}$ respectively the difference error from the attack $\mathbf{d_a}$, i.e.,

$$\text{Attack success} = \begin{cases} 1 & \text{if } \mathrm{HW}(\mathbf{d_a}) \leq t \vee \mathrm{HW}(\mathbf{e_a}) \leq t \\ 0 & \text{otherwise.} \end{cases} \tag{7.35}$$

Fig. 7.4 shows results for the improved DPA based on traces from a behavioral simulation. The number of traces is increased for different noise levels and the averaged attack success among the ten repetitions is provided. Independent of the code parameters, the attack success increases for an increasing number of traces. This is expected for a CPA attack as the influence of noise is averaged out with an increasing number of traces. The noise level that allows for a successful attack is similar across the different code parameters; a noise level of $\sigma \leq 10$ allows for attacking the selected code successfully with more than 1000 traces. Even for increased noise conditions of $\sigma = 50$, the attack success is above 0.5 for the available 10000 traces, i.e., an attack succeeds in more than half of the attempts.

An exception is the $(15, 5, 3)$ BCH code that can also be attacked at higher noise levels with a probability above 0.5 in many cases as shown in Fig. 7.4a. First, the lower code rate of the $(15, 5, 3)$ code adds additional redundancy in the codeword that is beneficial for the attacker as relatively more errors can be corrected. Second, note that the number of different manipulation patterns are $\binom{15}{3} = 455$ according to Section 7.4.1, i.e., patterns are used more than once for $N_t > 455$. In other words, traces

[7] The noise level $\sigma$ for load/store instructions has been measured on different microcontroller architectures for a noisy Hamming Weight (HW) model $\mathrm{HW}(a) + \mathcal{N}(0, \sigma)$ of an intermediate $a$ [113]. On an 8-bit microcontroller (XMEGA 128D), for $\sigma = 0.5$ the model matches the practical measurements and for an 32-bit microcontroller (STM32F405) the standard deviation varies from $\sigma = 0.4$ to $\sigma = 3.0$, where averaging over 10 traces reduces the noise level to $\sigma = 0.2 - 1.3$. While the noise levels have been obtained on microcontrollers and a HW model has been used, the values provide a reference point.




**(a)** (15, 5, 3) BCH code

**(b)** (63, 36, 5) BCH code

**(c)** (127, 64, 10) BCH code

**(d)** (255, 131, 18) BCH code

**Figure 7.4** Average attack success on simulated traces for the improved DPA: Noise vs. traces.

with the same manipulation pattern occur more than once, and (similar to averaging) the SNR is improved.

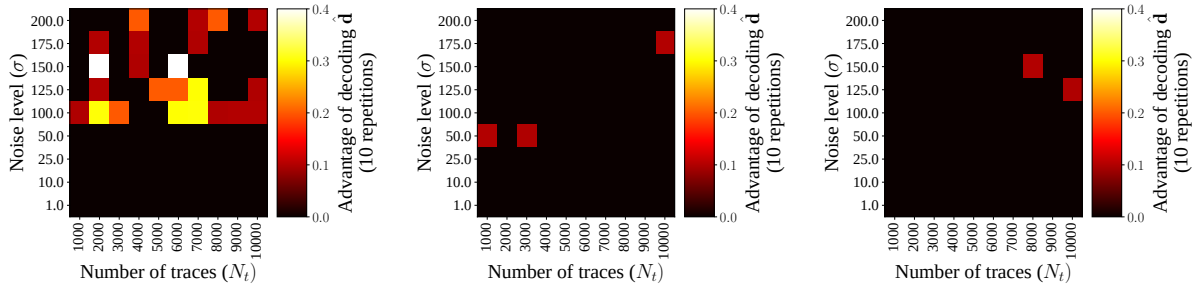
Finally, for the (15, 5, 3) code there is a higher number of cases where $\mathrm{HW}(\mathbf{d_a}) \leq t$, while $\mathrm{HW}(\mathbf{e_a}) > t$, i.e., where decoding the difference codeword $\hat{\mathbf{d}}$ provides benefits over decoding the codeword $\hat{\mathbf{c}}$ directly. The effect is depicted in Fig. 7.5 in terms of the number of repetitions where the difference codeword $\hat{\mathbf{d}}$ yields a successful attack, but the codeword $\hat{\mathbf{c}}$ does not.[8] For the (15, 5, 3) BCH code in Fig. 7.5a, the use of the difference codeword is advantageous in particular for higher noise levels $\sigma > 50$. The region in Fig. 7.5a, where decoding $\hat{\mathbf{d}}$ provides an advantage, corresponds to region at which the (15, 5, 3) code can still be attacked in Fig. 7.4a. In contrast, in Figs. 7.5b and 7.5c for the (63, 36, 5) and the (127, 64, 10) BCH code, attacking the difference codeword provides an advantage in only a few cases.

For all codes, the cases are in regions at the border between successful and unsuccessful attacks in Fig. 7.4. Thus, the approach proposed in Section 7.4.4 provides indeed an advantage compared to only decoding the codeword $\hat{\mathbf{c}}$ under certain conditions. In particular for smaller codes and higher noise levels, results can be improved compared to using only the codeword $\hat{\mathbf{c}}$.

### 7.5.3 On the Limits of Horizontal SCA of BCH Codes

This section provides simulation results for the correlation-based HSCA attack from Section 7.4.2. The same four BCH codes as in Section 7.5.2 are compared. As the horizontal attack uses only a single trace for the attack, increasing the number of traces per codeword, as for the DPA, does not provide any benefit. Instead, for each code 100 codewords are randomly chosen and the attack is carried out on the different codewords to reduce the influence of single codewords.

[8] The (255, 131, 18) code is not depicted in Fig. 7.5 as there is no case in which the difference codeword provides a benefit.




**(a)** $(15, 5, 3)$ BCH code **(b)** $(63, 36, 5)$ BCH code **(c)** $(127, 64, 10)$ BCH code

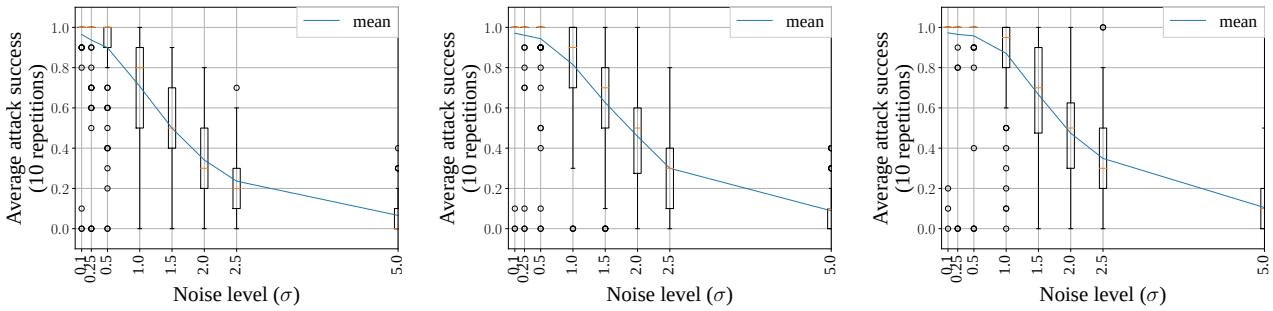
**Figure 7.5** Improved DPA on simulated traces: Advantage of decoding the difference codeword $\hat{\mathbf{d}}$ compared to decoding only the retrieved codeword $\hat{\mathbf{c}}$.

For the simulated power consumption of the entire BCH code, the horizontal attack does not succeed. Therefore, *the simulation takes only the LFSRs of the syndrome computation into account*, i.e., switching noise from the input buffer, the syndrome logic, the BMA and the Chien search is excluded. Eight noise levels are simulated ranging from $\sigma = 0.1$ to $\sigma = 5$, which is lower compared to the noise used for the improved DPA. For the smaller $(15, 5, 3)$ and $(63, 36, 5)$ codes ten repetitions per codeword and noise level are simulated. As the time for the attacks increases for larger codes, for the $(127, 64, 10)$ and $(255, 131, 18)$ codes only a single repetition is simulated per codeword and noise level. The HSCA attack has a degree of freedom, namely the hypothesis length. For the simulations, three different choices $m \in \{8, 10, 12\}$ are considered. Larger hypothesis length come at the cost of an increased computation time and memory consumption as the size of the hypothesis matrix $\mathbf{H}$ grows exponentially with the hypothesis length $m$. Therefore, the choice of the hypothesis length is practically limited by the computational resources, and for the results a trade-off in favor of simulating multiple codewords is made.

Figs. 7.6 to 7.9 show the simulation results for the HSCA. The attack success according to Eq. (7.35), averaged over the different codewords, is depicted for an increasing noise level. For simulations that use multiple repetitions per noise levels, box plots indicate the variation of the average attack success across the repetitions. The orange marker is the median value and the boxes span from the first to the third quartile of the data, i.e., half of the values lie within the Inter-Quartile Range (IQR) marked by the boxes. The circles indicate outliers that extend from the box by more than 1.5× of the IQR marked by the whiskers.

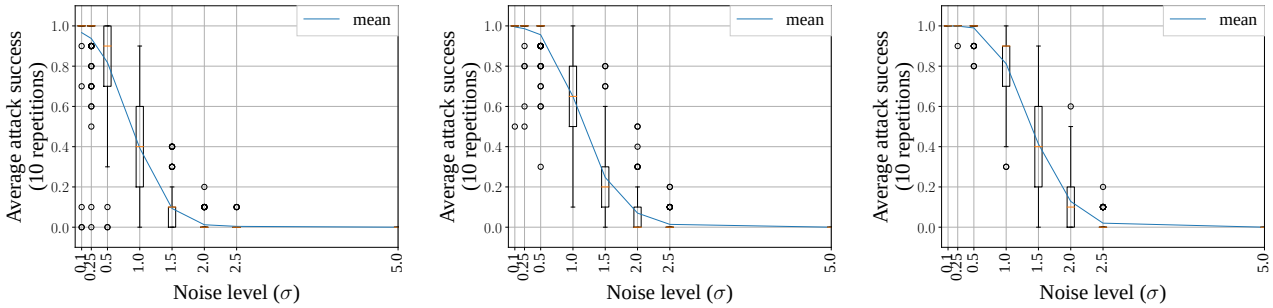
For all codes, increasing the hypothesis length $m$ improves the attack success. This is expected as more clock cycles are included into the hypothesis. Consequently, the correlation includes more points in time, and is more robust against noise. For noise levels $\sigma > 2.5$, the horizontal attack is only successful for some codewords of the $(15, 5, 3)$ BCH code in Fig. 7.6. An attack success above 0.75 is reached for $\sigma \leq 1.0$ with $m = 12$ for the $(15, 5, 3)$ and the $(63, 36, 5)$ codes in Figs. 7.6 and 7.7. For the $(127, 64, 10)$ code in Fig. 7.8 and the $(255, 131, 18)$ code in Fig. 7.9, an attack success above 0.75 is reached for $\sigma \leq 1.5$ with $m = 12$. The results indicate that the attack is slightly more robust for larger codes. A possible reason is that larger codes contain more LFSRs for the syndrome computation, i.e., the number of modeled registers is larger compared to smaller codes. Consequently, it is less likely that different candidates lead to similar hypothetical power consumption and the probability that the first bit of the selected candidate is wrong is decreased.

The results show that HSCA on BCH codes are possible in theory. The attack succeeds for power traces from a behavioral simulation with low additive noise, but only if switching noise is neglected. For a practical application, an attacker would need to resolve only the syndrome computation LFSRs, e.g., by localized EM measurements. At the same time the noise of the measurements must be very low. These strong requirements emphasize that the HSCA faces severe limitations regarding its practical application.



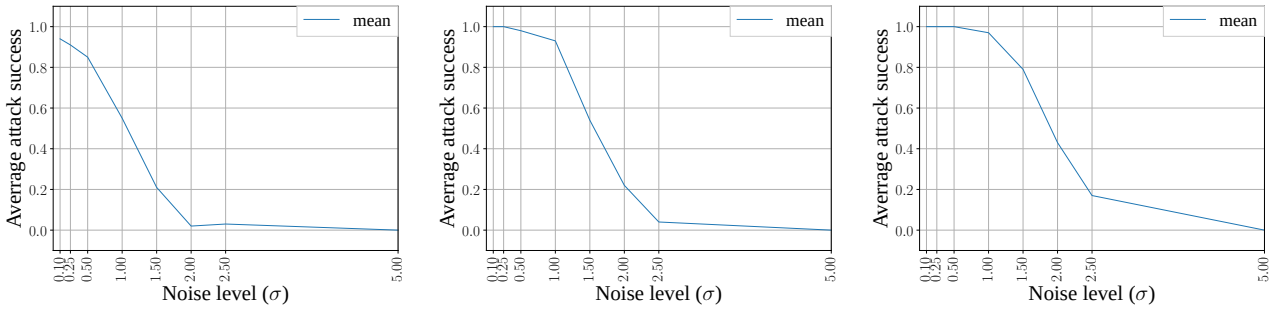

**(a)** $m = 8$ **(b)** $m = 10$ **(c)** $m = 12$

**Figure 7.6** Results for the simulated horizontal SCA for the $(15, 5, 3)$ BCH code.



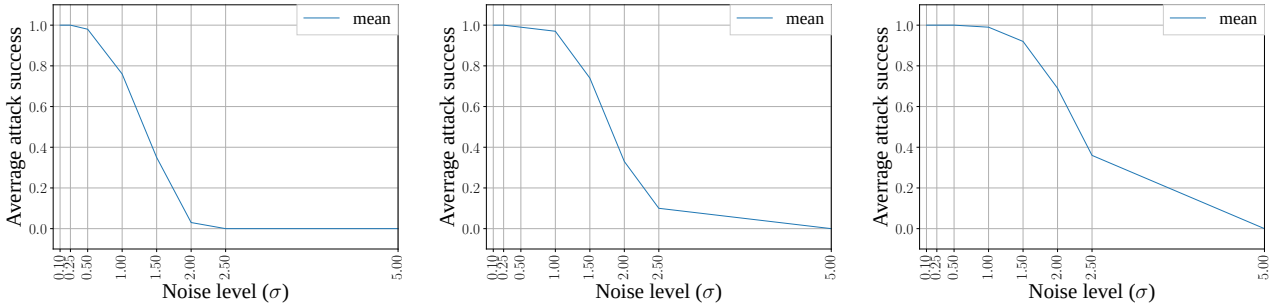

**(a)** $m = 8$ **(b)** $m = 10$ **(c)** $m = 12$

**Figure 7.7** Results for the simulated horizontal SCA for the $(63, 36, 5)$ BCH code.




**(a)** $m = 8$ **(b)** $m = 10$ **(c)** $m = 12$

**Figure 7.8** Results for the simulated horizontal SCA for the $(127, 64, 10)$ BCH code.




**(a)** $m = 8$ **(b)** $m = 10$ **(c)** $m = 12$

**Figure 7.9** Results for the simulated horizontal SCA for the $(255, 131, 18)$ BCH code.

## 7.6 Conclusion

The improved DPA attack from Section 7.4.1, using the syndrome power model and a modified helper data manipulation, is able to attack stand-alone BCH codes without the need for a concatenated repetition code. Manipulating only $t$ helper data bits per trace ensures that the decoding result is not affected and the manipulations cannot be detected by the decoder. Using the difference codeword as proposed in Section 7.4.4, can improve the vertical attack in particular for small codes and higher noise levels. The improved DPA attack is robust for a noise level of $\sigma \leq 50$ with up to $N_t = 10000$ traces and for a noise level of $\sigma \leq 10$ with only $N_t = 1000$ traces.

The results for the HSCA attack show that the approach faces practical limitations: The simulated power measurements include only the LFSRs of the syndrome computation, i.e., switching noise is neglected. This would imply that an attacker is able to resolve the relevant part of the circuit with high precision measurements, e.g., by using localized EM measurements. Furthermore, the noise levels that can be handled by the attack are considerably lower compared to the noise levels for the improved DPA attack. In other words, even if a spatial resolution could be achieved that allows for measuring only the LFSRs, additionally measurements with a high SNR would be required.

Summing up, if helper data manipulation is possible, countermeasures such as codeword masking [49] should be implemented to prevent DPA attacks. The HSCA on single traces without helper data manipulation is theoretically possible, but practically limited.

# 8 Conclusion

This thesis demonstrated the threat of SCA of PUF primitives by practical attacks on the TERO PUF and the Loop PUF showing that ring-based PUF primitives can be attacked by analyzing the duration or frequency of the oscillations. The comparison of unprotected ring-based PUF primitives regarding the difficulty of SCA showed that primitives deriving the secret from the stable frequency of a ring, such as Loop and RO PUF, are particularly prone to SCA. For primitives, whose secret is not directly related to the frequency, e.g., TERO, BR, and TBR PUF, the complexity of the measurement methods and their required precision for SCA increases significantly. While the unprotected Loop PUF is an easy target for SCA, the use of a single counter enables efficient protection mechanism, and a set of countermeasures has been proposed for the Loop PUF primitive. First, for sign-based bit derivation temporal masking, i.e., randomizing the order of two challenges, prevents an attacker from obtaining information about the corresponding secret bit. Deriving the randomness from the Loop PUF allows for constructing a self-secured PUF primitive. Second, methods for protecting amplitude-based bit derivation have been investigated with a particular focus on the TMHD scheme. Lightweight countermeasures based on the randomization of the challenge order do not provide a sufficient level of protection. However, the modified ICLooPUF primitive hides spectral side-channel leakage by interleaving the measurements of two challenges.

The different approaches to protect the Loop PUF highlight that the design of countermeasures for PUFs has to consider the combination of the PUF primitive and its bit derivation method. Countermeasures that focus only on the primitive may not be suitable if the bit derivation method is changed, potentially putting the security of the key generation at risk. However, countermeasure mechanisms can be transferred to other primitives as exemplified for temporal masking to the RO PUF and TERO PUF.

Finally, SCA attacks on the error correction for PUF-based key generation have been investigated. The widely used class of BCH codes has been evaluated regarding the possibility of vertical and horizontal SCA attacks. An improved DPA attack allows for attacking stand-alone BCH codes independent of the concatenation with a repetition code by including the LFSRs of the syndrome computation in the power model. Independent of the code parameters, it is robust against noise. Additionally, exploiting the properties of linear cyclic codes by using the difference codeword can increase the attack success under certain conditions. Therefore, DPA attacks have to be considered a serious threat that requires countermeasures. Restricting the write access to the helper data impedes vertical SCA attacks, such as DPA. However, the proposed correlation-based horizontal SCA attack combines different points in time of a single trace and does not rely on helper data manipulation. The results from simulated power measurements show that the attack requires a perfect match of the power measurements and the attacker's power model. Additionally, its application is limited to very low levels of noise. Thus, while theoretically feasible, horizontal SCA faces practical limitations in terms of the needed local resolution and the SNR.

Summing up, this thesis showed that countermeasures are required to enable a protected key generation from ring-based PUFs. Countermeasures for the Loop PUF have been proposed that protect sign-based and amplitude-based bit derivation and can partially be transferred to other primitives. Furthermore, DPA attacks of BCH codes used to correct errors in the PUF response have been extended to stand-alone codes, and the possibility and limitations of horizontal SCA have been analyzed.

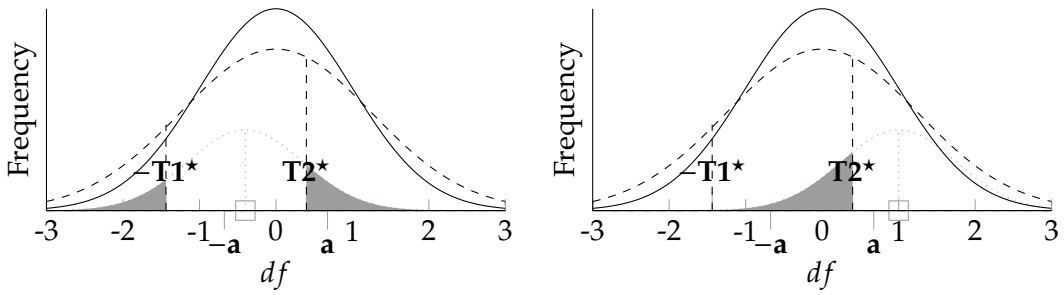
# A Analysis of the TMHD Scheme: With Helper Data Access and Without Temporal Masking

This section provides details for the attack success of an attacker, who attacks a Loop PUF without the temporal masking countermeasure and the TMHD bit derivation, and has helper data access. The additional helper data information improves the attack success compared to the attack in Section 5.2.3, which highlights that the TMHD scheme without further protection enables SCA. Note that without temporal masking, the frequency difference $df$ would be revealed independently of the helper data scheme, i.e., the following analysis is rather of theoretical interest. However, the results show that the reliability information of the TMHD can also be exploited by the attacker and improves the attack compared to the scenario without helper data knowledge.

Figs. A.1a and A.1b 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. 5.1a and 5.1b, the gray 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. The attack success probability is formalized in the following.

Assuming metric $M1$ and the value $df > a$ during enrollment the actual PUF bit is $r_C = 0$ according to Eqs. (2.4) and (2.5). The attacker will know that $M1$ is the metric but any observed value $-T1^\star \leq df'_C < T2^\star$ is decoded as $\hat{r}_C = 1 \neq r_C$. Now for $df^\star \sim \mathcal{N}(df, \sigma_{adv.})$, the probability for this event is

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




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

**Figure A.1** Visualization of the attack failure for attacker with helper data knowledge. As an example metric $M1$ is used, but no temporal masking is effective.

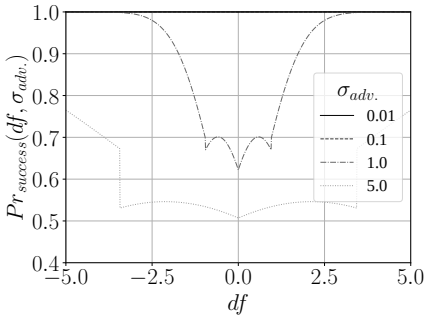



**Figure A.2** Helper data/no temporal masking: Simulation of the attack success probability for different levels of attacker noise $\sigma_{adv.}$.

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

$$\begin{aligned} P_2(df, \sigma_{adv.}) &= Pr[\hat{r}_C \neq r_C | w_C^{bd} = M1, -a \leq df \leq 0] \\ &= \int_{-\infty}^{-T1^\star} \phi^\star \left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star + \int_{T2^\star}^{\infty} \phi^\star \left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star. \end{aligned} \tag{A.2}$$

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

$$\begin{aligned} P_3(df, \sigma_{adv.}) &= Pr[\hat{r}_C \neq r_C | w_C^{bd} = M2, df < -a] \\ &= \int_{-T2^\star}^{T1^\star} \phi^\star \left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star, \end{aligned} \tag{A.3}$$

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

$$\begin{aligned} P_4(df, \sigma_{adv.}) &= Pr[\hat{r}_C \neq r_C | w_C^{bd} = M2, 0 < df \leq a] \\ &= \int_{-\infty}^{-T2^\star} \phi^\star \left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star + \int_{T1^\star}^{\infty} \phi^\star \left(df^\star; df, \sigma_{adv.}\right) \mathrm{d}df^\star. \end{aligned} \tag{A.4}$$

From the probabilities in Eqs. (A.1) to (A.4), which define the entire support of $df$, the overall success probability to recover a PUF bit is given by

$$Pr_{success}(df, \sigma_{adv.}) = 1 - \sum_{i=1}^{4} P_i(df, \sigma_{adv.}). \tag{A.5}$$

Fig. A.2 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 $r'_C$ is wrong. The attacker faces the highest uncertainty for values of $df$ close to the boundary between $\hat{r} = 0$ and $\hat{r} = 1$. 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.

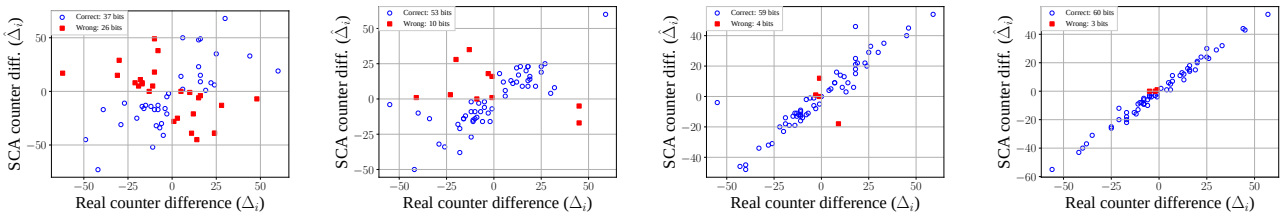
[1]Note: For the standard normal distribution $\mu = 0$, $\sigma = 1$, the resulting value are $|\pm T1| = 0.31863936$, $|\pm a| = 0.67448975$ and $|\pm T2| = 1.15034938$. Depending on $\sigma$, the value are scaled accordingly. Notably the points that define the octiles are not equidistant.

# B Additional Results for On-Chip Power Analysis of the Loop PUF

This appendix provides further details for the results in Section 4.2.3. First, Appendix B.1 shows the comparison of estimated and real counter values for Figs. 4.11 to 4.13. Second, Appendix B.2 provides the spectra from the TDC sensor data.

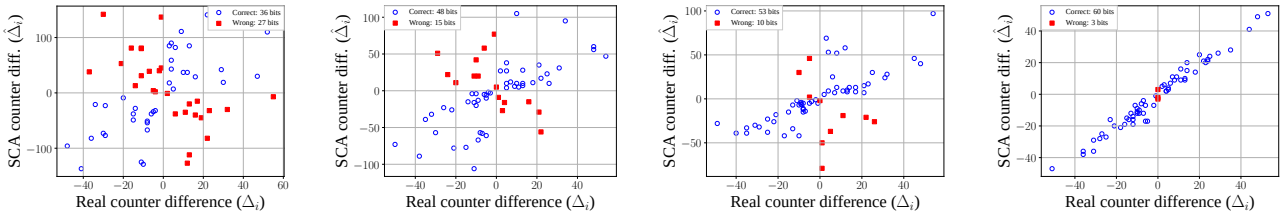
## B.1 Comparison of Estimated and Real Counter Values

This section provides some additional figures that show the relationship between the actual counter difference on the device and counter difference retrieved by SCA for the different scenarios. In Figs. B.1 to B.4 the results for the CW305 with $clk_{tdc} = 40\,\text{MHz}$ from Fig. 4.11 are depicted. Similarly, Figs. B.5 to B.8 depict the match for the Basys3 with $clk_{tdc} = 30\,\text{MHz}$ from Fig. 4.12. Finally, Figs. B.9 and B.10 provide details for the comparison of and remote SCA of the CW305 with $clk_{tdc} = 16\,\text{MHz}$ and classical SCA, corresponding to the results in Fig. B.4.



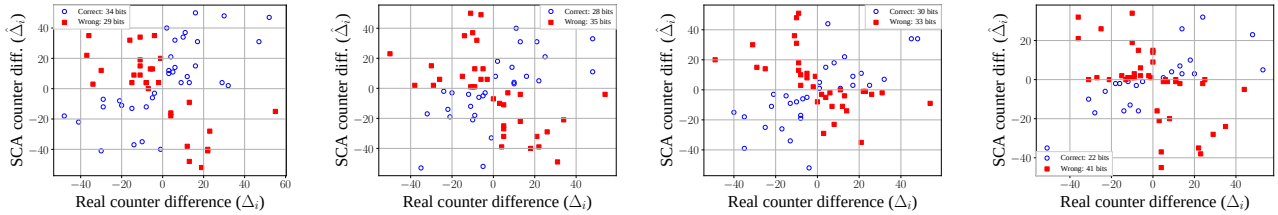

**(a)** 1 repetition **(b)** 50 repetitions **(c)** 100 repetitions **(d)** 500 repetitions

**Figure B.1** Comparison of real counter value and SCA results from counter FoIs on the CW305 for $clk_{tdc} = 40\,\text{MHz}$ and close placement – corresponding to Fig. 4.11a.



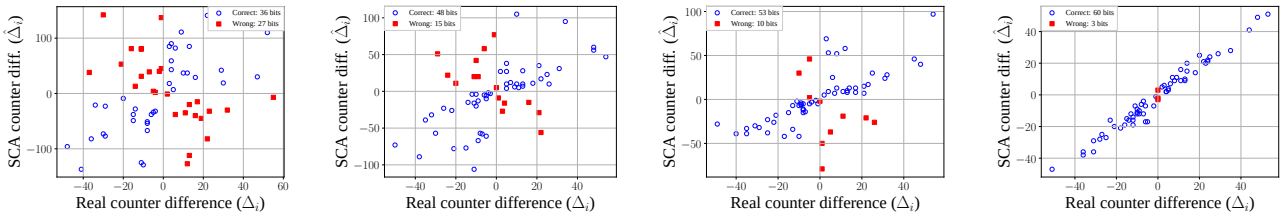

**(a)** 1 repetition **(b)** 50 repetitions **(c)** 100 repetitions **(d)** 500 repetitions

**Figure B.2** Comparison of real counter value and SCA results from oscillator FoIs on the CW305 for $clk_{tdc} = 40\,\text{MHz}$ and close placement – corresponding to Fig. 4.11b.



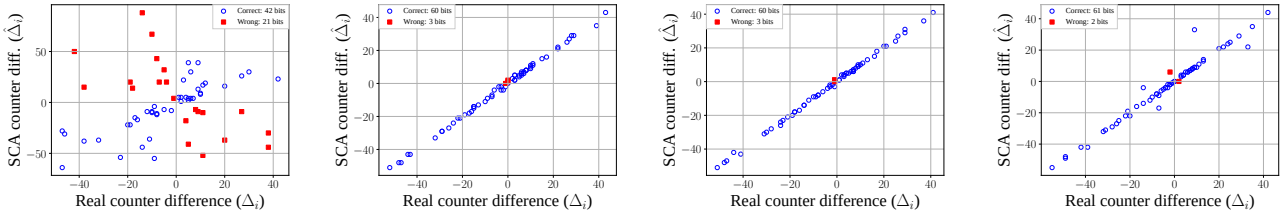

**(a)** 1 repetition **(b)** 50 repetitions **(c)** 100 repetitions **(d)** 500 repetitions

**Figure B.3** Comparison of real counter value and SCA results from counter FoIs on the CW305 for $clk_{tdc}$ = 40 MHz and separate placement and – corresponding to Fig. 4.11c.



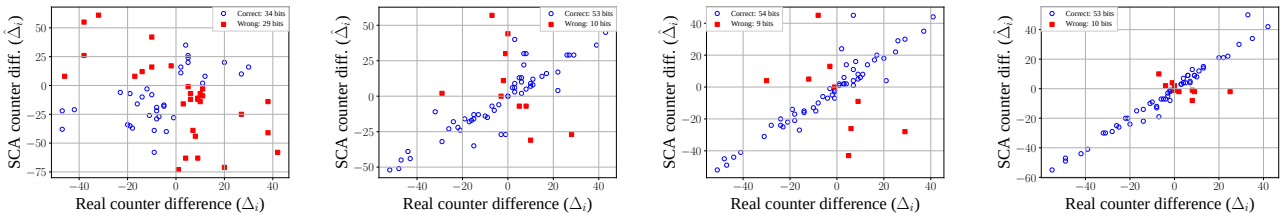

**(a)** 1 repetition **(b)** 50 repetitions **(c)** 100 repetitions **(d)** 500 repetitions

**Figure B.4** Comparison of real counter value and SCA results from oscillator FoIs on the CW305 for $clk_{tdc}$ = 40 MHz and separate placement – corresponding to Fig. 4.11d.



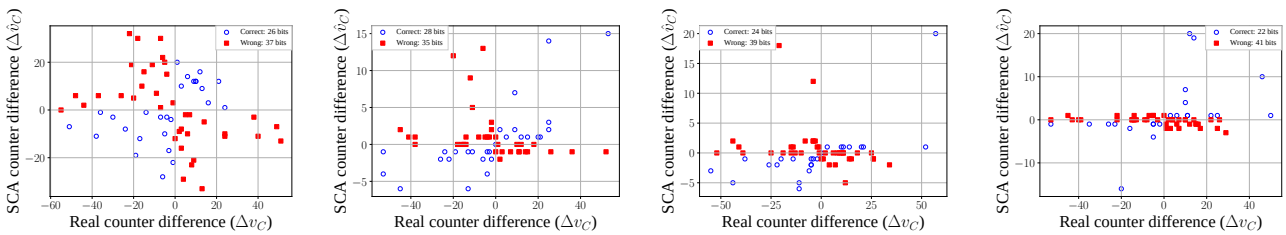

**(a)** 1 repetition **(b)** 50 repetitions **(c)** 100 repetitions **(d)** 500 repetitions

**Figure B.5** Comparison of real counter value and SCA results from counter FoIs on the Basys3 for $clk_{tdc}$ = 30 MHz and close placement – corresponding to Fig. 4.12a.



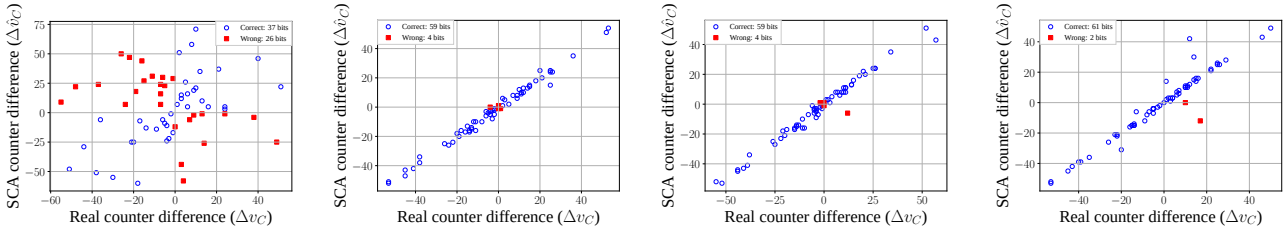

**(a)** 1 repetition **(b)** 50 repetitions **(c)** 100 repetitions **(d)** 500 repetitions

**Figure B.6** Comparison of real counter value and SCA results from oscillator FoIs on the Basys3 for $clk_{tdc}$ = 30 MHz and close placement – corresponding to Fig. 4.12b.



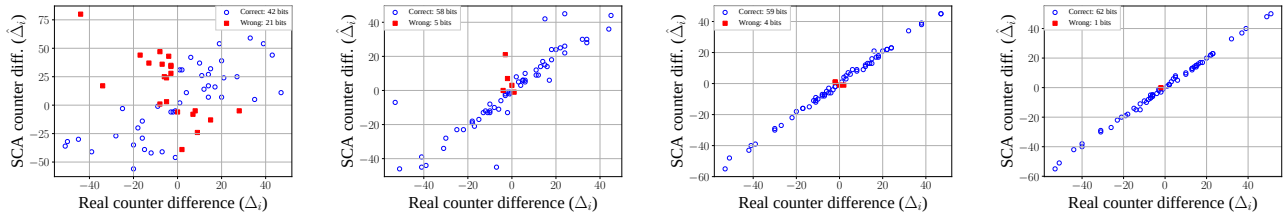

**(a)** 1 repetition **(b)** 50 repetitions **(c)** 100 repetitions **(d)** 500 repetitions

**Figure B.7** Comparison of real counter value and SCA results from counter FoIs on the Basys3 for $clk_{tdc}$ = 30 MHz and separate placement – corresponding to Fig. 4.12c.



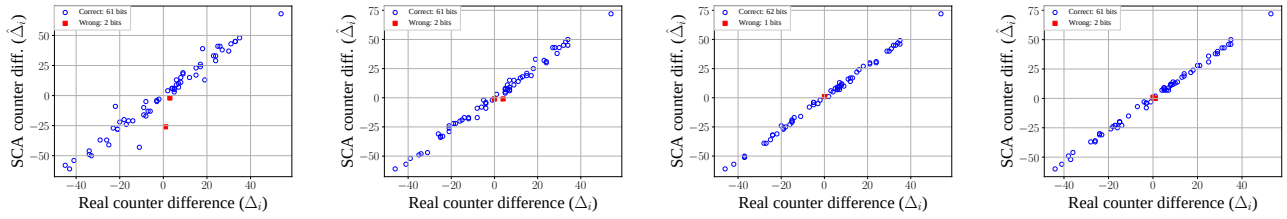

**(a)** 1 repetition **(b)** 50 repetitions **(c)** 100 repetitions **(d)** 500 repetitions

**Figure B.8** Comparison of real counter value and SCA results from oscillator FoIs on the Basys3 for $clk_{tdc}$ = 30 MHz and separate placement – corresponding to Fig. 4.12d.




**(a)** 1 repetition **(b)** 10 repetition **(c)** 50 repetitions **(d)** 100 repetitions

**Figure B.9** Comparison of real counter value and SCA results from counter FoIs on the CW305 for $clk_{tdc}$ = 16 MHz and close placement.




**(a)** 1 repetition **(b)** 2 repetition **(c)** 5 repetitions **(d)** 10 repetitions

**Figure B.10** Comparison of real counter value and SCA results from counter FoIs on the CW305 for classical SCA from power measurements.

## B.2 Frequency Spectrum Plots

Fig. B.11 depicts the averaged spectra for 63 different challenges from the Basys3 board, separate placement, and with $clk_{tdc} = 30\,\text{MHz}$. The spectrum for each challenges is averaged over 500 repetitions to increase the SNR and smoothing of the spectrum is applied to ease peak detection. In addition to the entire frequency range, a zoom to the FoI range used for the attack is shown. The frequency peaks in the FoI range span about 30 kHz. Consequently, the peaks are separate enough, such that automatic peak detection can distinguish them. This is also indicated by the comparison of real and estimated counter values in Fig. B.8.

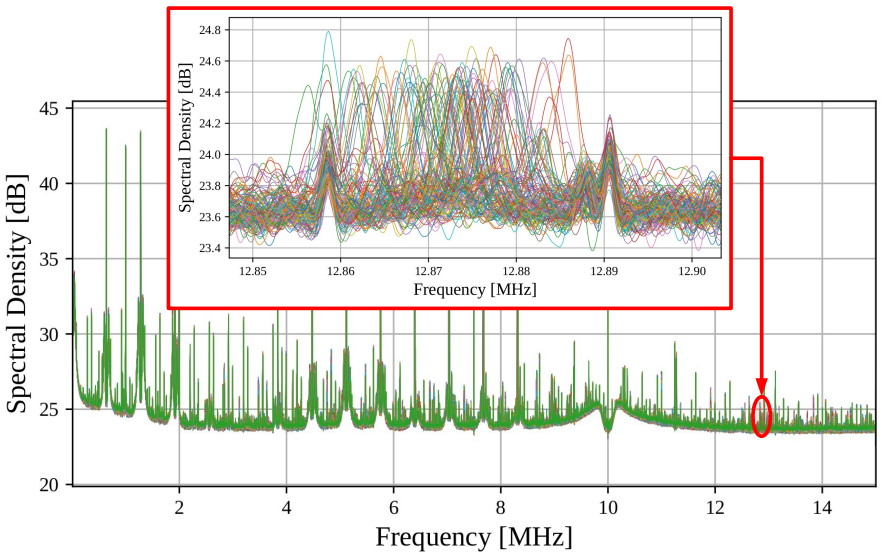



**Figure B.11** Frequency spectrum averaged for 500 repetitions on the Basys3 board, $clk_{tdc} = 30\,\text{MHz}$ and separate placement.

# C Theoretical Leakage Behavior of the ICLooPUF

This appendix provides a detailed derivation of the theoretical frequency leakage of the ICLooPUF in Eq. (5.20). First, a solution for infinite signals is provided in Appendix C.1, and subsequently extended to time-limited signals in Appendix C.2.

## C.1 Infinite Time Signals

The signal generated by the Interleaved Loop-PUF can be described as a series of interleaved sine waves, with periods $T_C$ and $T_{C'}$, i.e.,

$$g(t) = g_1(t) + g_2(t) \tag{C.1}$$

with

$$g_1(t) = \sum_{k=1}^{N} \sin\left(\frac{2\pi}{T_C}\left(t - (k-1)\left(T_C + T_{\neg C}\right)\right)\right) \cdot \Theta\left(\frac{\left(t - \left((k-1)\left(T_C + T_{\neg C}\right) + \frac{T_C}{2}\right)\right)}{T_C}\right) \tag{C.2}$$

$$g_2(t) = \sum_{k=1}^{N-1} \sin\left(\frac{2\pi}{T_{\neg C}}\left(t - \left(kT_C + (k-1)\,T_{\neg C}\right)\right)\right) \cdot \Theta\left(\frac{\left(t - \left(k\left(T_C + T_{\neg C}\right)\right) + \frac{T_{\neg C}}{2}\right)}{T_{\neg C}}\right).$$

In order to simulate the theoretical spectral leakage behavior, the frequency representation of $g_1(t)$ and $g_2(t)$ is of interest.

The Fourier series, which is a periodic function with period $P$ consisting of weighted sinusoids, can approximate an arbitrary function $s(t)$ by

$$s_N(t) = \sum_{n=-N}^{N} c_n \cdot e^{j\frac{2\pi}{P}nt}, \quad c_n = \frac{1}{P}\int_P s(t) \cdot e^{-j\frac{2\pi}{P}nt}\, dt. \tag{C.3}$$

The signal $g(t)$ from Eq. (C.1) is periodic with $P = T_C + T_{C'}$, such that the coefficients $c_n$ are obtained as

$$c_n = \frac{1}{T_C + T_{C'}} \int_0^{T_C + T_{C'}} g(t) \cdot e^{-j\frac{2\pi}{T_C + T_{C'}}nt}\, dt \tag{C.4}$$

$$= \frac{1}{T_C + T_{C'}} \left[\int_0^{T_C} \sin\left(\frac{2\pi}{T_C}t\right) \cdot e^{-j\frac{2\pi}{T_C + T_{C'}}nt}\, dt + \int_{T_C}^{T_C + T_{C'}} \sin\left(\frac{2\pi}{T_{C'}}(t - T_C)\right) \cdot e^{-j\frac{2\pi}{T_C + T_{C'}}nt}\, dt\right] \tag{C.5}$$

From the product rule $(uv)' = u'v + v'u$, it follows that $\int_a^b u'v = [uv]_a^b - \int_a^b v'u$. With $u' = e^{-jdt}$ and $v = \sin\left(\frac{2\pi}{c}(t-t')\right)$ the integral is solved as ($b > a$, and $a, b, c, d, t' \geq 0$)

$$\int_a^b e^{-jdt} \sin\left(\frac{2\pi}{c}(t-t')\right) dt = \underbrace{\left[\sin\left(\frac{2\pi}{c}(t-t')\right)\frac{1}{-jd}\cdot e^{-jdt}\right]_a^b}_{A} - \int_a^b \frac{1}{-jd}\cdot e^{-jdt}\frac{2\pi}{c}\cdot\cos\left(\frac{2\pi}{c}(t-t')\right)dt$$

$$= A + \frac{2\pi}{jdc}\int_a^b \underbrace{e^{-jdt}}_{\hat{u}'}\cdot\underbrace{\cos\left(\frac{2\pi}{c}(t-t')\right)}_{\hat{v}}dt$$

$$= A + \frac{2\pi}{jdc}\left(\left[\cos\left(\frac{2\pi}{c}(t-t')\right)\frac{1}{-jd}\cdot e^{-jdt}\right]_a^b - \int_a^b \frac{2\pi}{jdc}\cdot e^{-jdt}\cdot\sin\left(\frac{2\pi}{c}(t-t')\right)dt\right)$$

$$= A + \frac{2\pi}{d^2c}\left[\cos\left(\frac{2\pi}{c}(t-t')\right)\cdot e^{-jdt}\right]_a^b + \left(\frac{2\pi}{dc}\right)^2\int_a^b e^{-jdt}\cdot\sin\left(\frac{2\pi}{c}(t-t')\right)dt$$

Grouping the integrals, and assuming $a = t'$, $b = kc + t'$, $k \in \mathbb{N}$, i.e., integration is done over a period of the harmonic functions, the solution can be simplified:

$$\int_a^b e^{-jdt}\sin\left(\frac{2\pi}{c}(t-t')\right)dt = \left(1-\left(\frac{2\pi}{dc}\right)^2\right)^{-1}\left(\underbrace{\left[\sin\left(\frac{2\pi}{c}(t-t')\right)\frac{j}{d}\cdot e^{-jdt}\right]_a^b}_{=0} + \frac{2\pi}{d^2c}\left[\cos\left(\frac{2\pi}{c}(t-t')\right)\cdot e^{-jdt}\right]_a^b\right) \tag{C.6}$$

$$= \left(1-\left(\frac{2\pi}{dc}\right)^2\right)^{-1}\frac{2\pi}{d^2c}\left(e^{-jdb} - e^{-jda}\right) = \frac{2\pi c}{d^2c^2 - 4\pi^2}\left(e^{-jdb} - e^{-jda}\right) \tag{C.7}$$

With $d = \frac{2\pi n}{T_C + T_{C'}}$, $a = 0$, $b = T_C$, $c = T_C$, and $t' = 0$ the first integral in Eq. (C.5) solves to

$$\int_0^{T_C} e^{-j\frac{2\pi n}{T_C+T_{C'}}t}\sin\left(\frac{2\pi}{T_C}t\right)dt = \frac{2\pi T_C}{\left(\frac{2\pi n}{T_C+T_{C'}}\right)^2 T_C^2 - 4\pi^2}\left(e^{-j\frac{2\pi n}{T_C+T_{C'}}T_C} - 1\right) \tag{C.8}$$

$$= \frac{T_C}{\frac{2\pi n^2}{(T_C+T_{C'})^2}T_C^2 - 2\pi}\left(e^{-j\frac{2\pi n}{T_C+T_{C'}}T_C} - 1\right) = \frac{T_C}{2\pi}\frac{1}{\frac{n^2}{(T_C+T_{C'})^2}T_C^2 - 1}\left(e^{-j\frac{2\pi n}{T_C+T_{C'}}T_C} - 1\right) \tag{C.9}$$

With $d = \frac{2\pi n}{T_C+T_{C'}}$, $a = T_C$, $b = T_C + T_{C'}$, $c = T_{C'}$, and $t' = T_C$ the second integral in Eq. (C.5) solves to

$$\int_{T_C}^{T_C+T_{C'}} e^{-j\frac{2\pi n}{T_C+T_{C'}}t}\sin\left(\frac{2\pi}{T_{C'}}(t-T_C)\right)dt = \frac{2\pi T_{C'}}{\left(\frac{2\pi n}{T_C+T_{C'}}\right)^2 T_{C'}^2 - 4\pi^2}\left(e^{-j\frac{2\pi n}{T_C+T_{C'}}(T_C+T_{C'})} - e^{-j\frac{2\pi n}{T_C+T_{C'}}T_C}\right) \tag{C.10}$$

$$= \frac{T_{C'}}{2\pi}\frac{1}{\frac{n^2}{(T_C+T_{C'})^2}T_{C'}^2 - 1}\left(1 - e^{-j\frac{2\pi n}{T_C+T_{C'}}T_C}\right) \tag{C.11}$$

Substituting the expressions from Eqs. (C.8) and (C.10) into Eq. (C.3) results in the Fourier series approximation of the signal:

$$
\begin{aligned}
s_N(t) &= \sum_{n=-N}^{N} c_n \cdot e^{j\frac{2\pi}{T_C+T_{C'}}nt} \\
&= \sum_{n=-N}^{N} \frac{1}{T_C+T_{C'}} \left( \frac{T_C}{2\pi} \frac{1}{\frac{n^2}{(T_C+T_{C'})^2}T_C^2 - 1} \left( e^{-j\frac{2\pi n}{T_C+T_{C'}}T_C} - 1 \right) + \frac{T_{C'}}{2\pi} \frac{1}{\frac{n^2}{(T_C+T_{C'})^2}T_{C'}^2 - 1} \left( 1 - e^{-j\frac{2\pi n}{T_C+T_{C'}}T_C} \right) \right) e^{j\frac{2\pi}{T_C+T_{C'}}nt} \\
&= \frac{1}{2\pi} \sum_{n=-N}^{N} \frac{1}{T_C+T_{C'}} \left( \frac{T_C}{\frac{n^2}{(T_C+T_{C'})^2}T_C^2 - 1} \left( e^{-j\frac{2\pi n}{T_C+T_{C'}}T_C} - 1 \right) + \frac{T_{C'}}{\frac{n^2}{(T_C+T_{C'})^2}T_{C'}^2 - 1} \left( 1 - e^{-j\frac{2\pi n}{T_C+T_{C'}}T_C} \right) \right) e^{j\frac{2\pi}{T_C+T_{C'}}nt} \\
&= \frac{T_C+T_{C'}}{2\pi} \sum_{n=-N}^{N} \left( \underbrace{\frac{T_C}{n^2T_C^2 - (T_C+T_{C'})^2}}_{:=\alpha(n)} \left( e^{-j\frac{2\pi n}{T_C+T_{C'}}T_C} - 1 \right) + \underbrace{\frac{T_{C'}}{n^2T_{C'}^2 - (T_C+T_{C'})^2}}_{:=\beta(n)} \left( 1 - e^{-j\frac{2\pi n}{T_C+T_{C'}}T_C} \right) \right) e^{j\frac{2\pi}{T_C+T_{C'}}nt} \\
&= \frac{T_C+T_{C'}}{2\pi} \sum_{n=-N}^{N} (\alpha(n) - \beta(n))\, e^{j\frac{2\pi n}{T_C+T_{C'}}(t-T_C)} + (\beta(n) - \alpha(n))\, e^{j\frac{2\pi}{T_C+T_{C'}}nt}
\end{aligned}
\tag{C.12}
$$

Finally, we can transform Eq. (C.12) to the frequency domain using the conversion of complex harmonic functions $e^{jat} \circ\!\!-\!\!\bullet \delta\left(f - \frac{a}{2\pi}\right)$ and the time shift property $h(t-t_0) \circ\!\!-\!\!\bullet e^{j2\pi t_0 f} H(f)$, where $h(t)$ is the time domain function and $H(f)$ the corresponding Fourier transform:

$$
\begin{aligned}
S(f) &= \int_{-\infty}^{\infty} s_N(t) \cdot e^{-j2\pi ft}\, dt \\
&= \frac{T_C+T_{C'}}{2\pi} \int_{-\infty}^{\infty} \sum_{n=-N}^{N} \left( (\alpha(n) - \beta(n))\, e^{j\frac{2\pi n}{T_C+T_{C'}}(t-T_C)} + (\beta(n) - \alpha(n))\, e^{j\frac{2\pi}{T_C+T_{C'}}nt} \right) \cdot e^{-j2\pi ft}\, dt \\
&= \frac{T_C+T_{C'}}{2\pi} \sum_{n=-N}^{N} (\alpha(n) - \beta(n))\, e^{j2\pi fT_C}\, \delta\left( f - \frac{n}{T_C+T_{C'}} \right) + (\beta(n) - \alpha(n))\, \delta\left( f - \frac{n}{T_C+T_{C'}} \right) \\
&= \frac{T_C+T_{C'}}{2\pi} \sum_{n=-N}^{N} \left[ (\alpha(n) - \beta(n))\, e^{j2\pi fT_C} + (\beta(n) - \alpha(n)) \right] \delta\left( f - \frac{n}{T_C+T_{C'}} \right)
\end{aligned}
\tag{C.13}
$$

So, we can conclude that the frequency of the interleaved signal is only present for multiples of the average frequency $f = \frac{n}{T_C+T_{C'}}$ with decreasing amplitudes for $|n| \to \infty$ as $\alpha(n), \beta(n) \sim \frac{1}{n^2}$.

The remaining question is whether it is possible to learn something from the amplitudes of the different harmonics $n$:

$$
\begin{aligned}
\left| S\left( \frac{n}{T_C+T_{C'}} \right) \right| &= \frac{T_C+T_{C'}}{2\pi} \left| \left( \frac{T_C}{n^2T_C^2 - (T_C+T_{C'})^2} - \frac{T_{C'}}{n^2T_{C'}^2 - (T_C+T_{C'})^2} \right) e^{j2\pi\frac{n}{T_C+T_{C'}}T_C} \right. \\
&\qquad \left. + \frac{T_{C'}}{n^2T_{C'}^2 - (T_C+T_{C'})^2} - \frac{T_C}{n^2T_C^2 - (T_C+T_{C'})^2} \right| \\
&= \frac{T_C+T_{C'}}{2\pi} \left| \frac{T_C}{n^2T_C^2 - (T_C+T_{C'})^2} \left( e^{j2\pi\frac{n}{T_C+T_{C'}}T_C} - 1 \right) + \frac{T_{C'}}{n^2T_{C'}^2 - (T_C+T_{C'})^2} \left( 1 - e^{j2\pi\frac{n}{T_C+T_{C'}}T_C} \right) \right|
\end{aligned}
$$

Setting $T_{C'} := \gamma T_C,\ \gamma > 0$ the spectral amplitude can be expressed as the relative period of the interleaved signals:

$$\begin{aligned}|S(n,\gamma)| &= \frac{(1+\gamma)T_C}{2\pi}\left|\frac{T_C}{n^2T_C^2-(1+\gamma)^2T_C^2}\left(e^{j2\pi\frac{n}{(1+\gamma)T_C}T_C}-1\right)+\frac{\gamma T_C}{n^2\gamma^2T_C^2-(1+\gamma)^2T_C^2}\left(1-e^{j2\pi\frac{n}{(1+\gamma)T_C}T_C}\right)\right| \\ &= \frac{(1+\gamma)}{2\pi}\left|\frac{1}{n^2-(1+\gamma)^2}\left(e^{j2\pi\frac{n}{(1+\gamma)}}-1\right)+\frac{\gamma}{n^2\gamma^2-(1+\gamma)^2}\left(1-e^{j2\pi\frac{n}{(1+\gamma)}}\right)\right| \end{aligned} \tag{C.14}$$

Eq. (C.14) allows for evaluating whether certain harmonics $n$ of the interleaved signal are particulary present and whether and how the amplitude changes for different relations $\gamma$ of the periods of the interleaved signals.

## C.2 Extension for Time-Limited Signals

The result in Eq. (C.13) represents the ideal spectrum for an infinite signal $s_N(t)$. However, in reality the signal $s_N(t)$ is limited to $N_{max}$ clock cycles of the period $P = T_C + T_{C'}$. For the time limited signal $s_N(t)\cdot\Theta\left(\frac{t}{N_{max}\cdot P}\right)$, the convolutional properties of the Dirac-function $\delta(\cdot)$ lead to $e^{jat}\cdot\Theta(bt)\circ\!\!-\!\!\bullet\delta\left(f-\frac{a}{2\pi}\right)*\frac{1}{|b|}\operatorname{sinc}\left(\frac{f}{b}\right)=\frac{1}{|b|}\operatorname{sinc}\left(\frac{f-\frac{a}{2\pi}}{b}\right)$, with $\operatorname{sinc}(f)=\sin(\pi f)/(\pi f)$. In other words, the Dirac-function in Eq. (C.13) is replaced by a sinc-function

$$\begin{aligned}\frac{1}{|b|}\operatorname{sinc}\left(\frac{f-\frac{a}{2\pi}}{b}\right) &= |N_{max}(T_C+T_{C'})|\cdot\operatorname{sinc}\left(N_{max}(T_C+T_{C'})\left(f-\frac{n}{T_C+T_{C'}}\right)\right) \\ &= |N_{max}(1+\gamma)T_C|\cdot\operatorname{sinc}\left(N_{max}(1+\gamma)T_C\left(f-\frac{n}{(1+\gamma)T_C}\right)\right)\end{aligned}$$

In contrast to Eqs. (C.13) and (C.14), where discrete frequencies corresponding to the harmonics can be evaluated, using the sinc-function the mixture of different $n$ has to be considered for the entire spectrum:

$$\begin{aligned}|S(f)| = \Bigg|&\frac{(1+\gamma)^2T_CN_{max}}{2\pi}\sum_{n=-N}^{N}\left[\frac{1}{n^2-(1+\gamma)^2}\left(e^{j2\pi\frac{n}{(1+\gamma)}}-1\right)+\frac{\gamma}{n^2\gamma^2-(1+\gamma)^2}\left(1-e^{j2\pi\frac{n}{(1+\gamma)}}\right)\right] \\ &\cdot\operatorname{sinc}\left(N_{max}(1+\gamma)T_C\left(f-\frac{n}{(1+\gamma)T_C}\right)\right)\Bigg|\end{aligned}$$

# D Details of the Investigated BCH Codes

This section provides the details of the BCH codes that are evaluated in Chapter 7. The minimal polynomials $\phi_i(x)$ define the LFSR configurations used for the improved power model. Furthermore, the Galois field and its defining irreducible polynomial as well as the generator polynomial that results from the Least Common Multiple (LCM) of the minimal polynomials are provided.

## D.1 (15, 5, 3) BCH Code

Galois field $GF\left(2^4\right)$ with the irreducible polynomial $x^4 + x + 1$, generator polynomial: $g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$ with the minimal polynomials:

$$\begin{aligned} \phi_1(x) = \phi_2(x) = \phi_4(x) &= x^4 + x + 1 \\ \phi_3(x) = \phi_6(x) &= x^4 + x^3 + x^2 + x + 1 \\ \phi_5(x) &= x^2 + x + 1 \end{aligned}$$

## D.2 (63, 36, 5) BCH Code

Galois field $GF\left(2^6\right)$ with the irreducible polynomial $x^6 + x + 1$, generator polynomial: $g(x) = x^{27} + x^{22} + x^{21} + x^{19} + x^{18} + x^{17} + x^{15} + x^8 + x^4 + x + 1$ with the minimal polynomials:

$$\begin{aligned} \phi_1(x) = \phi_2(x) = \phi_4(x) = \phi_8(x) &= x^6 + x + 1 \\ \phi_3(x) = \phi_6(x) = \phi_9(x) &= x^6 + x^4 + x^2 + x + 1 \\ \phi_5(x) = \phi_{10}(x) &= x^6 + x^5 + x^2 + x + 1 \\ \phi_7(x) &= x^6 + x^3 + 1 \\ \phi_9(x) &= x^3 + x^2 + 1 \end{aligned}$$

## D.3 (127, 64, 10) BCH Code

Galois field $GF\left(2^7\right)$ with the irreducible polynomial $x^7 + x^3 + 1$, generator polynomial: $g(x) = x^{63} + x^{61} + x^{56} + x^{55} + x^{53} + x^{51} + x^{49} + x^{48} + x^{47} + x^{40} + x^{38} + x^{36} + x^{35} + x^{33} + x^{32} + x^{31} + x^{30} + x^{26} + x^{25} + x^{24} + x^{23} + x^{22} + x^{21} + x^{19} + x^{18} + x^{15} + x^5 + x^2 + 1$ with the minimal polynomials:

$$\begin{aligned}
\phi_1(x) = \phi_2(x) = \phi_4(x) = \phi_8(x) = \phi_{16}(x) &= x^7 + x^3 + 1 \\
\phi_3(x) = \phi_6(x) = \phi_{12}(x) &= x^7 + x^3 + x^2 + x + 1 \\
\phi_5(x) = \phi_{10}(x) = \phi_{20}(x) &= x^7 + x^4 + x^3 + x^2 + 1 \\
\phi_7(x) = \phi_{14}(x) &= x^7 + x^6 + x^5 + x^4 + x^2 + x + 1 \\
\phi_9(x) = \phi_{17}(x) = \phi_{18}(x) &= x^7 + x^5 + x^4 + x^3 + x^2 + x + 1 \\
\phi_{11}(x) &= x^7 + x^6 + x^4 + x^2 + 1 \\
\phi_{13}(x) &= x^7 + x + 1 \\
\phi_{15}(x) &= x^7 + x^6 + x^5 + x^3 + x^2 + x + 1 \\
\phi_{19}(x) &= x^7 + x^6 + x^3 + x + 1
\end{aligned}$$

## D.4 (255, 131, 18) BCH Code

Galois field $GF\left(2^8\right)$ with the irreducible polynomial $x^8 + x^4 + x^3 + x^2 + 1$, generator polynomial $g(x) = x^{124} + x^{120} + x^{119} + x^{117} + x^{116} + x^{115} + x^{114} + x^{111} + x^{109} + x^{108} + x^{106} + x^{105} + x^{103} + x^{102} + x^{99} + x^{98} + x^{95} + x^{94} + x^{93} + x^{90} + x^{89} + x^{87} + x^{84} + x^{78} + x^{77} + x^{75} + x^{72} + x^{70} + x^{68} + x^{67} + x^{63} + x^{61} + x^{59} + x^{57} + x^{52} + x^{50} + x^{49} + x^{48} + x^{47} + x^{46} + x^{45} + x^{44} + x^{41} + x^{37} + x^{33} + x^{32} + x^{28} + x^{22} + x^{20} + x^{15} + x^{14} + x^{13} + x^{11} + x^9 + x^8 + x^5 + x^4 + x^3 + 1$ with the minimal polynomials:

$$\begin{aligned}
\phi_1(x) = \phi_2(x) = \phi_4(x) = \phi_8(x) = \phi_{16}(x) = \phi_{32}(x) &= x^8 + x^4 + x^3 + x^2 + 1 \\
\phi_3(x) = \phi_6(x) = \phi_{12}(x) = \phi_{24}(x) &= x^8 + x^6 + x^5 + x^4 + x^2 + x + 1 \\
\phi_5(x) = \phi_{10}(x) = \phi_{20}(x) &= x^8 + x^7 + x^6 + x^5 + x^4 + x + 1 \\
\phi_7(x) = \phi_{14}(x) = \phi_{28}(x) &= x^8 + x^6 + x^5 + x^3 + 1 \\
\phi_9(x) = \phi_{18}(x) = \phi_{33}(x) = \phi_{36}(x) &= x^8 + x^7 + x^5 + x^4 + x^3 + x^2 + 1 \\
\phi_{11}(x) = \phi_{22}(x) &= x^8 + x^7 + x^6 + x^5 + x^2 + x + 1 \\
\phi_{13}(x) = \phi_{26}(x) &= x^8 + x^5 + x^3 + x + 1 \\
\phi_{15}(x) = \phi_{30}(x) &= x^8 + x^7 + x^6 + x^4 + x^2 + x + 1 \\
\phi_{17}(x) = \phi_{34}(x) &= x^4 + x + 1 \\
\phi_{19}(x) &= x^8 + x^6 + x^5 + x^2 + 1 \\
\phi_{21}(x) &= x^8 + x^7 + x^3 + x + 1 \\
\phi_{23}(x) &= x^8 + x^6 + x^5 + x + 1 \\
\phi_{25}(x) = \phi_{35}(x) &= x^8 + x^4 + x^3 + x + 1 \\
\phi_{27}(x) &= x^8 + x^5 + x^4 + x^3 + x^2 + x + 1 \\
\phi_{29}(x) &= x^8 + x^7 + x^3 + x^2 + 1 \\
\phi_{31}(x) &= x^8 + x^5 + x^3 + x^2 + 1
\end{aligned}$$

# Bibliography


[1] Y. Lee, B. Karpinskyy, Y. Choi, K.-M. Ahn, Y. Kim, J. Park, S. Noh, J. Kang, J. Shin, J. Park, Y. Chung, and J. Shin, "Samsung physically unclonable function (SAMPUF™) and its integration with samsung security system," in *2021 IEEE Custom Integrated Circuits Conference (CICC)*, 2021, pp. 1–7.

[2] M.-Y. Wu, T.-H. Yang, L.-C. Chen, C.-C. Lin, H.-C. Hu, F.-Y. Su, C.-M. Wang, J. P.-H. Huang, H.-M. Chen, C. C.-H. Lu, E. C.-S. Yang, and R. S.-J. Shen, "A PUF scheme using competing oxide rupture with bit error rate approaching zero," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 2018, pp. 130–132.

[3] K. K.-H. Chuang, H.-M. Chen, M.-Y. Wu, E. C.-S. Yang, and C. C.-H. Hsu, "Quantum tunneling PUF: A chip fingerprint for hardware security," in *2021 International Symposium on VLSI Technology, Systems and Applications (VLSI-TSA)*, 2021, pp. 1–2.

[4] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *Proceedings of the 6th ACM Conference on Computer and Communications Security*, ser. CCS '99. ACM, 1999, pp. 28–36.

[5] S. Katzenbeisser, Ü. Kocabaş, V. Rožić, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann, "PUFs: Myth, fact or busted? a security evaluation of physically unclonable functions (PUFs) cast in silicon," in *Cryptographic Hardware and Embedded Systems – CHES 2012*, E. Prouff and P. Schaumont, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 283–301.

[6] A. Wild, G. T. Becker, and T. Güneysu, "A fair and comprehensive large-scale analysis of oscillation-based PUFs for FPGAs," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, Sept 2017, pp. 1–7.

[7] R. Hesselbarth, F. Wilde, C. Gu, and N. Hanley, "Large scale RO PUF analysis over slice type, evaluation time and temperature on 28nm Xilinx FPGAs," in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, April 2018, pp. 126–133.

[8] R. Maes and V. van der Leest, "Countering the effects of silicon aging on SRAM PUFs," in *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2014, pp. 148–153.

[9] A. Maiti and P. Schaumont, "The impact of aging on a physical unclonable function," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 9, pp. 1854–1864, 2014.

[10] R. Maes, "An accurate probabilistic reliability model for silicon PUFs," in *Cryptographic Hardware and Embedded Systems - CHES 2013*, G. Bertoni and J.-S. Coron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 73–89.

[11] M. Pehl, M. Hiller, and G. Sigl, "Secret key generation for physical unclonable functions," in *Information Theoretic Security and Privacy of Information Systems*, R. F. Schaefer, H. Boche, A. Khisti, and H. V. Poor, Eds. Cambridge University Press, 2017, p. 362–389.

[12] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Advances in Cryptology - EUROCRYPT 2004*, C. Cachin and J. L. Camenisch, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 523–540.

[13] G. Davida, Y. Frankel, and B. Matt, "On enabling secure applications through off-line biometric identification," in *Proceedings. 1998 IEEE Symposium on Security and Privacy (Cat. No.98CB36186)*, 1998, pp. 148–157.

[14] M. Hiller, M.-D. M. Yu, and M. Pehl, "Systematic low leakage coding for physical unclonable functions," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 155–166.

[15] M.-D. Yu and S. Devadas, "Secure and robust error correction for physical unclonable functions," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 48–65, 2010.

[16] M. Hiller, D. Merli, F. Stumpf, and G. Sigl, "Complementary IBS: Application specific error correction for PUFs," in *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*, 2012, pp. 1–6.

[17] M. Hiller, M. Weiner, L. Rodrigues Lima, M. Birkner, and G. Sigl, "Breaking through fixed PUF block limitations with differential sequence coding and convolutional codes," in *Proceedings of the 3rd International Workshop on Trustworthy Embedded Devices*, ser. TrustED '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 43–54.

[18] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM conference on Computer and communications security (CCS '02)*, 2002, pp. 148–160.

[19] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the Design Automation Conference, (DAC '07). 44th ACM/IEEE*, 2007, pp. 9–14.

[20] Z. Cherif, J. Danger, S. Guilley, and L. Bossuet, "An easy-to-design PUF based on a single oscillator: The loop PUF," in *2012 15th Euromicro Conference on Digital System Design*, Sep. 2012, pp. 156–162.

[21] L. Tebelmann, J.-L. Danger, and M. Pehl, "Interleaved challenge loop PUF: A highly side-channel protected oscillator-based PUF," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 12, pp. 5121–5134, 2022.

[22] M. Varchola, M. Drutarovsky, and V. Fischer, "New universal element with integrated PUF and TRNG capability," in *2013 International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, Dec 2013, pp. 1–6.

[23] L. Bossuet, X. T. Ngo, Z. Cherif, and V. Fischer, "A PUF based on a transient effect ring oscillator and insensitive to locking phenomenon," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 1, pp. 30–36, March 2014.

[24] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, and U. Rührmair, "The bistable ring PUF: A new architecture for strong physical unclonable functions," in *2011 IEEE International Symposium on Hardware-Oriented Security and Trust*, June 2011, pp. 134–141.

[25] ——, "Characterization of the bistable ring PUF," in *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2012, pp. 1459–1462.

[26] R. Hesselbarth and G. Sigl, "Fast and reliable PUF response evaluation from unsettled bistable rings," in *2016 Euromicro Conference on Digital System Design (DSD)*, Aug 2016, pp. 82–90.

[27] M. Riedel, "Understanding TBR PUF: State trajectory analysis on an FPGA array," Master's thesis, Technical University of Munich, February 2018.

[28] F. Wilde, B. M. Gammel, and M. Pehl, "Spatial correlation analysis on physical unclonable functions," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 6, pp. 1468–1480, 2018.

[29] R. Maes, A. Van Herrewege, and I. Verbauwhede, "PUFKY: A fully functional PUF-based cryptographic key generator," in *Cryptographic Hardware and Embedded Systems – CHES 2012*, ser. Lecture Notes in Computer Science, E. Prouff and P. Schaumont, Eds., vol. 7428, 2012, p. 302–319.

[30] C.-E. D. Yin and G. Qu, "LISA: Maximizing RO PUF's secret extraction," in *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2010, pp. 100–105.

[31] O. Rioul, P. Solé, S. Guilley, and J.-L. Danger, "On the entropy of physically unclonable functions," in *2016 IEEE International Symposium on Information Theory (ISIT)*, 2016, pp. 2928–2932.

[32] A. Cherkaoui, L. Bossuet, and C. Marchand, "Design, evaluation, and optimization of physical unclonable functions based on transient effect ring oscillators," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1291–1305, June 2016.

[33] C. Marchand, L. Bossuet, and A. Cherkaoui, "Design and characterization of the TERO-PUF on SRAM FPGAs," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, July 2016, pp. 134–139.

[34] C. Marchand, L. Bossuet, U. Mureddu, N. Bochard, A. Cherkaoui, and V. Fischer, "Implementation and characterization of a physical unclonable function for IoT: A case study with the TERO-PUF," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 97–109, Jan 2018.

[35] M. Varchola and M. Drutarovsky, "New high entropy element for FPGA based true random number generators," in *Cryptographic Hardware and Embedded Systems, CHES 2010*, S. Mangard and F.-X. Standaert, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 351–365.

[36] F. Ganji, S. Tajik, F. Fäßler, and J.-P. Seifert, "Having no mathematical model may not secure PUFs," *Journal of Cryptographic Engineering*, vol. 7, no. 2, pp. 113–128, Jun 2017.

[37] D. Schuster and R. Hesselbarth, "Evaluation of bistable ring PUFs using single layer neural networks," in *Trust and Trustworthy Computing*, T. Holz and S. Ioannidis, Eds. Cham: Springer International Publishing, 2014, pp. 101–109.

[38] B. Škorić, P. Tuyls, and W. Ophey, "Robust key extraction from physical uncloneable functions," in *Applied Cryptography and Network Security*, J. Ioannidis, A. Keromytis, and M. Yung, Eds. Springer Berlin Heidelberg, 2005, pp. 407–422.

[39] F. Armknecht, R. Maes, A.-R. Sadeghi, B. Sunar, and P. Tuyls, "Memory leakage-resilient encryption based on physically unclonable functions," in *Advances in Cryptology – ASIACRYPT 2009*, M. Matsui, Ed. Springer Berlin Heidelberg, 2009, pp. 685–702.

[40] J. Danger, S. Guilley, and A. Schaub, "Two-metric helper data for highly robust and secure delay PUFs," in *2019 IEEE 8th International Workshop on Advances in Sensors and Interfaces (IWASI)*, June 2019, pp. 184–188.

[41] T. Stanko, F. Nur Andini, and B. Škorić, "Optimized quantization in zero leakage helper data systems," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1957–1966, 2017.

[42] V. Immler and K. Uppund, "New insights to key derivation for tamper-evident physical unclonable functions," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 3, pp. 30–65, May 2019.

[43] V. Immler, M. Hennig, L. Kürzinger, and G. Sigl, "Practical aspects of quantization and tamper-sensitivity for physically obfuscated keys," in *Proceedings of the Third Workshop on Cryptography and Security in Computing Systems*, ser. CS2 '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 13–18.

[44] C. Bösch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, and P. Tuyls, "Efficient helper data key extractor on FPGA," in *Cryptographic Hardware and Embedded Systems – CHES 2008. 10th international workshop. Washington, DC, USA, August 10 - 13, 2008. Proceedings*, E. Oswald and P. Rohatgi, Eds. Springer, 2008, pp. 181–197.

[45] S. Lin, *Error Control Coding*, 2nd ed., D. J. Costello, Ed. Pearson-Prentice Hall, 2004.

[46] Y. Choi, B. Karpinskyy, K.-M. Ahn, Y. Kim, S. Kwon, J. Park, Y. Lee, and M. Noh, "Physically unclonable function in 28nm FDSOI technology achieving high reliability for AEC-Q 100 grade 1 and ISO26262 ASIL-B," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2020, pp. 426–428.

[47] A. Van Herrewege, S. Katzenbeisser, R. Maes, R. Peeters, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann, "Reverse fuzzy extractors: Enabling lightweight mutual authentication for PUF-enabled RFIDs," in *Financial Cryptography and Data Security*, A. D. Keromytis, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 374–389.

[48] A. Van Herrewege and I. Verbauwhede, "Tiny application-specific programmable processor for BCH decoding," in *2012 International Symposium on System on Chip (SoC)*, Oct 2012, pp. 1–4.

[49] D. Merli, F. Stumpf, and G. Sigl, "Protecting PUF error correction by codeword masking," *IACR Cryptology ePrint Archive*, vol. 334, 2013.

[50] B. Jarvis and K. Gaj, "Selection of an error-correcting code for FPGA-based physical unclonable functions," in *2017 International Conference on Field Programmable Technology (ICFPT)*, Dec 2017, pp. 243–246.

[51] S. Mangard, *Power Analysis Attacks*, E. Oswald and T. Popp, Eds. Springer, 2007.

[52] C. Helfmeier, C. Boit, D. Nedospasov, and J. Seifert, "Cloning physically unclonable functions," in *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, June 2013, pp. 1–6.

[53] Y. Oren, A.-R. Sadeghi, and C. Wachsmann, "On the effectiveness of the remanence decay side-channel to clone memory-based PUFs," in *Cryptographic Hardware and Embedded Systems - CHES 2013*, G. Bertoni and J.-S. Coron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 107–125.

[54] S. Zeitouni, Y. Oren, C. Wachsmann, P. Koeberl, and A. Sadeghi, "Remanence decay side-channel: The PUF case," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1106–1116, June 2016.

[55] S. Tajik, E. Dietz, S. Frohmann, J.-P. Seifert, D. Nedospasov, C. Helfmeier, C. Boit, and H. Dittrich, "Physical characterization of arbiter PUFs," in *Cryptographic Hardware and Embedded Systems – CHES 2014*, L. Batina and M. Robshaw, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 493–509.

[56] H. Lohrke, S. Tajik, C. Boit, and J.-P. Seifert, "No place to hide: Contactless probing of secret data on FPGAs," in *Cryptographic Hardware and Embedded Systems – CHES 2016*, B. Gierlichs and A. Y. Poschmann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 147–167.

[57] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Semi-invasive EM attack on FPGA RO PUFs and countermeasures," in *6th Workshop on Embedded Systems Security (WESS'2011)*. ACM, Mar 2011.

[58] D. Merli, J. Heyszl, B. Heinz, D. Schuster, F. Stumpf, and G. Sigl, "Localized electromagnetic analysis of RO PUFs," in *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, June 2013, pp. 19–24.

[59] M. Shiozaki and T. Fujino, "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*, ser. ASHES'19. New York, NY, USA: ACM, 2019, pp. 13–21.

[60] P. Bayon, L. Bossuet, A. Aubert, and V. Fischer, "Electromagnetic analysis on ring oscillator-based true random number generators," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, May 2013, pp. 1954–1957.

[61] L. Sauvage, S. Guilley, and Y. Mathieu, "Electromagnetic radiations of FPGAs: High spatial resolution cartography and attack on a cryptographic module," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, no. 1, pp. 4:1–4:24, Mar. 2009.

[62] A. T. Markettos and S. W. Moore, "The frequency injection attack on ring-oscillator-based true random number generators," in *Cryptographic Hardware and Embedded Systems - CHES 2009*, C. Clavier and K. Gaj, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 317–331.

[63] P. Bayon, L. Bossuet, A. Aubert, V. Fischer, F. Poucheret, B. Robisson, and P. Maurine, "Contactless electromagnetic active attack on ring oscillator based true random number generator," in *Constructive Side-Channel Analysis and Secure Design*, W. Schindler and S. A. Huss, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 151–166.

[64] L. Tebelmann, M. Pehl, and V. Immler, "Side-channel analysis of the TERO PUF," in *Constructive Side-Channel Analysis and Secure Design*, I. Polian and M. Stöttinger, Eds. Cham: Springer International Publishing, 2019, pp. 43–60.

[65] U. Mureddu, B. Colombier, N. Bochard, L. Bossuet, and V. Fischer, "Transient effect ring oscillators leak too," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, July 2019, pp. 37–42.

[66] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Side-channel analysis of PUFs and fuzzy extractors," in *Trust and Trustworthy Computing*, ser. Lecture Notes in Computer Science, J. M. McCune, B. Balacheff, A. Perrig, A.-R. Sadeghi, A. Sasse, and Y. Beres, Eds. Springer Berlin Heidelberg, 2011, no. 6740, pp. 33–47.

[67] D. Karakoyunlu and B. Sunar, "Differential template attacks on PUF enabled cryptographic devices," *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2010.

[68] L. Tebelmann, M. Pehl, and G. Sigl, "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*, ser. ASHES '17. New York, NY, USA: ACM, 2017, pp. 43–52.

[69] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Helper data algorithms for PUF-based key generation: Overview and analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 889–902, June 2015.

[70] J. Delvaux and I. Verbauwhede, "Key-recovery attacks on various RO PUF constructions via helper data manipulation," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2014, pp. 1–6.

[71] ——, "Attacking PUF-based pattern matching key generators via helper data manipulation," in *Topics in Cryptology – CT-RSA 2014*, ser. Lecture Notes in Computer Science, J. Benaloh, Ed. Springer International Publishing, 2014, no. 8366, pp. 106–131.

[72] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith, "Secure remote authentication using biometric data," in *Advances in Cryptology – EUROCRYPT 2005*, R. Cramer, Ed. Springer Berlin Heidelberg, 2005, pp. 147–163.

[73] G. T. Becker, "Robust fuzzy extractors and helper data manipulation attacks revisited: Theory versus practice," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 5, pp. 783–795, 2019.

[74] ——, "The gap between promise and reality: On the insecurity of XOR arbiter PUFs," in *Cryptographic Hardware and Embedded Systems – CHES 2015*, T. Güneysu and H. Handschuh, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 535–555.

[75] F. Ganji, *On the Learnability of Physically Unclonable Functions*. Springer International Publishing, 2018.

[76] G. T. Becker and R. Kumar, "Active and passive side-channel attacks on delay based PUF designs," Cryptology ePrint Archive, Report 2014/287, 2014.

[77] E. Strieder, C. Frisch, and M. Pehl, "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*, vol. 2021, no. 2, pp. 1–36, Feb. 2021.

[78] J. Ruchti, M. Gruber, and M. Pehl, "When the decoder has to look twice: Glitching a PUF error correction," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, no. 3, p. 26–70, 2022-06.

[79] V. Immler, R. Specht, and F. Unterstein, "Your rails cannot hide from localized EM: how dual-rail logic fails on FPGAs," in *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, 2017, pp. 403–424.

[80] F. Unterstein, J. Heyszl, F. De Santis, and R. Specht, "Dissecting leakage resilient PRFs with multivariate localized EM attacks," in *Constructive Side-Channel Analysis and Secure Design*, S. Guilley, Ed. Cham: Springer International Publishing, 2017, pp. 34–49.

[81] L. Hars, "Random number generation based on oscillatory metastability in ring circuits," Cryptology ePrint Archive, Paper 2011/637.

[82] P. Haddad, V. Fischer, F. Bernard, and J. Nicolai, "A physical approach for stochastic modeling of TERO-based TRNG," in *Cryptographic Hardware and Embedded Systems – CHES 2015*, T. Güneysu and H. Handschuh, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 357–372.

[83] F. Bernard, P. Haddad, V. Fischer, and J. Nicolai, "From physical to stochastic modeling of a TERO-based TRNG," *Journal of Cryptology*, vol. 32, no. 2, pp. 435–458, 2019.

[84] J. Delvaux, "Refutation and redesign of a physical model of TERO-based TRNGs and PUFs," Cryptology ePrint Archive, Report 2019/810, 2019.

[85] L. Tebelmann, J.-L. Danger, and M. Pehl, "Self-secured PUF: Protecting the loop PUF by masking," in *Constructive Side-Channel Analysis and Secure Design*, G. M. Bertoni and F. Regazzoni, Eds. Cham: Springer International Publishing, 2020, pp. 293–314.

[86] L. Tebelmann, M. Wettermann, and M. Pehl, "On-chip side-channel analysis of the loop PUF," in *Proceedings of the 2022 Workshop on Attacks and Solutions in Hardware Security*, ser. ASHES'22. New York, NY, USA: Association for Computing Machinery, 2022, p. 55–63.

[87] L. Feiten, K. Scheibler, B. Becker, and M. Sauer, "Using different LUT paths to increase area efficiencyof RO-PUFs on Altera FPGAs," in *TRUEDEVICE*, 2018.

[88] M. C. Martínez-Rodríguez, I. M. Delgado-Lozano, and B. B. Brumley, "SoK: Remote power analysis," in *The 16th International Conference on Availability, Reliability and Security*, ser. ARES 2021. New York, NY, USA: Association for Computing Machinery, 2021.

[89] D. R. E. Gnad, F. Oboril, S. Kiamehr, and M. B. Tahoori, "Analysis of transient voltage fluctuations in FPGAs," in *2016 International Conference on Field-Programmable Technology (FPT)*, 2016, pp. 12–19.

[90] M. Zhao and G. E. Suh, "FPGA-based remote power side-channel attacks," in *2018 IEEE Symposium on Security and Privacy (SP)*, May 2018, pp. 229–244.

[91] J. Gravellier, J. Dutertre, Y. Teglia, and P. Loubet-Moundi, "High-speed ring oscillator based sensors for remote side-channel attacks on FPGAs," in *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Dec 2019, pp. 1–8.

[92] F. Schellenberg, D. R. E. Gnad, A. Moradi, and M. B. Tahoori, "An inside job: Remote power analysis attacks on FPGAs," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2018, pp. 1111–1116.

[93] S. Moini, S. Tian, D. Holcomb, J. Szefer, and R. Tessier, "Remote power side-channel attacks on BNN accelerators in FPGAs," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2021, pp. 1639–1644.

[94] M. E. S. Elrabaa, M. Al-Asli, and M. Abu-Amara, "Secure computing enclaves using FPGAs," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 2, pp. 593–604, 2021.

[95] S. Tian, A. Krzywosz, I. Giechaskiel, and J. Szefer, "Cloud FPGA security with RO-based primitives," in *2020 International Conference on Field-Programmable Technology (ICFPT)*, 2020, pp. 154–158.

[96] R. Elnaggar, S. Ray, M. Sabbagh, B. Yuce, T. Wang, and J. Fung, "OPAL: On-the-go physical attack lab to evaluate power side-channel vulnerabilities on FPGAs," in *2021 IEEE Physical Assurance and Inspection of Electronics (PAINE)*, 2021, pp. 1–8.

[97] J. Wu, "Several key issues on implementing delay line based TDCs using FPGAs," *IEEE Transactions on Nuclear Science*, vol. 57, no. 3, pp. 1543–1548, 2010.

[98] M. Adamič and A. Trost, "A fast high-resolution time-to-digital converter implemented in a Zynq 7010 SoC," in *2019 Austrochip Workshop on Microelectronics (Austrochip)*, 2019, pp. 29–34.

[99] M. Wettermann, "Remote power analysis of the loop PUF using time-to-digital converters," Master's thesis, Technical University of Munich, April 2022.

[100] Y. Wang, J. Kuang, C. Liu, and Q. Cao, "A 3.9-ps RMS precision time-to-digital converter using ones-counter encoding scheme in a Kintex-7 FPGA," *IEEE Transactions on Nuclear Science*, vol. 64, no. 10, pp. 2713–2718, 2017.

[101] L. Tebelmann, U. Kühne, J.-L. Danger, and M. Pehl, "Analysis and protection of the two-metric helper data scheme," in *Constructive Side-Channel Analysis and Secure Design*, S. Bhasin and F. De Santis, Eds. Cham: Springer International Publishing, 2021, pp. 279–302.

[102] A. Maiti, V. Gunreddy, and P. Schaumont, "A systematic method to evaluate and compare the performance of physical unclonable functions," in *Embedded Systems Design with FPGAs*, P. Athanas, D. Pnevmatikatos, and N. Sklavos, Eds. New York, NY: Springer New York, 2013, pp. 245–267.

[103] V. Huber, "Design and side-channel evaluation of BCH code hardware implementations," Master's thesis, Technical University of Munich, May 2022.

[104] M. Brosch, "Horizontal side-channel analysis of error-correcting codes," Master's thesis, Technical University of Munich, November 2019.

[105] E. Jamro, "The design of a VHDL based synthesis tool for BCH codecs," Master's thesis, University of Huddersfield, September 1997.

[106] M. Yin, M. Xie, and B. Yi, "Optimized algorithms for binary BCH codes," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2013, pp. 1552–1555.

[107] W. Liu, J. Rho, and W. Sung, "Low-power high-throughput BCH error correction VLSI design for multi-level cell NAND flash memories," in *2006 IEEE Workshop on Signal Processing Systems Design and Implementation*, 2006, pp. 303–308.

[108] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99. CRYPTO 1999*, ser. Lecture Notes in Computer Science (LNCS), vol. 1666. Springer, Berlin, Heidelberg, 1999, pp. 388–397.

[109] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems - CHES 2004. CHES 2004*, ser. Lecture Notes in Computer Science (LNCS), vol. 3156. Springer, Berlin, Heidelberg, 2004, pp. 16–29.

[110] C. D. Walter, "Sliding windows succumbs to big mac attack," in *Cryptographic Hardware and Embedded Systems — CHES 2001*, Ç. K. Koç, D. Naccache, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 286–299.

[111] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil, "Horizontal correlation analysis on exponentiation," in *Information and Communications Security*, M. Soriano, S. Qing, and J. López, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 46–61.

[112] M. Gruber and G. Sigl, "TOFU - toggle count analysis made simple," Cryptology ePrint Archive, Paper 2022/129.

[113] M. J. Kannwischer, P. Pessl, and R. Primas, "Single-trace attacks on Keccak," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 3, pp. 243–268, Jun. 2020.

# Acronyms

| | |
|---|---|
| BCH | Bose-Chaudhuri-Hocquenghem |
| BER | Bit Error Rate |
| BMA | Berlekamp-Massey Algorithm |
| BR | Bistable Ring |
| BRAM | Block Random Access Memory |
| C-IBS | Complementary IBS |
| CDF | Cumulative Distribution Function |
| CGF | Code Generation Framework |
| CLB | Configurable Logic Block |
| CMOS | Complementary Metal-Oxide-Semiconductor |
| COFE | Code-Offset Fuzzy Extractor |
| CPA | Correlation Power Analysis |
| CRP | Challenge-Response Pair |
| CW305 | ChipWhisperer 305 Artix FPGA Target |
| DC | Direct Current |
| DPA | Differential Power Analysis |
| DSC | Differential Sequence Coding |
| ECC | Error-Correcting Code |
| EM | electromagnetic |
| EPQ | Equidistant Quantization |
| EPQ | Equiprobable Quantization |
| FCS | Fuzzy Commitment Scheme |
| FE | Fuzzy Extractor |
| FF | Flip-Flop |
| FFT | Fast Fourier Transform |
| FIA | Fault Injection Attacks |
| FIB | Focused Ion Beam |
| FiBM | Further optimized inversion-free Berlekamp-Massey |
| FoI | Frequency of Interest |
| FPGA | Field Programmable Gate Array |
| FSM | Finite State Machine |
| HDA | Helper Data Algorithm |
| HDF5 | Hierarchical Data Format 5 |
| HDMA | Helper Data Manipulation Attack |
| HSCA | Horizontal Side-Channel Analysis |
| IBS | Index-Based Syndrome |
| IC | Integrated Circuit |

| | |
|---|---|
| ICLooPUF | Interleaved Challenge Loop PUF |
| ILA | Integrated Logic Analyzer |
| IQR | Inter-Quartile Range |
| LCM | Least Common Multiple |
| LFSR | Linear Feedback Shift Register |
| LSB | Least Significant Bit |
| LUT | Lookup Table |
| ML | Machine Learning |
| MSB | Most Significant Bit |
| NVM | Non-Volatile Memory |
| PDF | Probability Density Function |
| PDN | Power Distribution Network |
| PE | Processing Element |
| PLL | Phase-Locked Loop |
| PSD | Power Spectral Density |
| PUF | Physical Unclonable Function |
| PUFKY | PUF-based cryptographic KeY generator design |
| RFE | Robust Fuzzy Extractor |
| RNG | Random Number Generator |
| RO | Ring Oscillator |
| RS | Reed-Solomon |
| RTL | Register-Transfer Level |
| SCA | Side-Channel Analysis |
| SLLC | Systematic Low Leakage Coding |
| SNR | Signal-to-Noise Ratio |
| SPA | Simple Power Analysis |
| SRAM | Static Random-Access Memory |
| STFT | Short-Time Fourier Transform |
| T-FF | Toggle Flip-Flop |
| TBR | Twisted Bistable Ring |
| Tcl | Tool command language |
| TDC | Time-to-Digital Converter |
| TERO | Transient Effect Ring Oscillator |
| TMHD | Two-Metric Helper Data |
| TOFU | TOggle Foul-Up |
| TRNG | True Random Number Generator |
| UART | Universal Asynchronous Receiver/Transmitter |
| VCD | Value Change Dump |
| VHDL | Very High Speed Integrated Circuit Hardware Description Language |

## Credits

Figs. 2.8 and 5.7 to 5.14, Tables 5.1, 5.7 to 5.9 and 6.1, and Section 5.5 (partially) reprinted, with permission, from Tebelmann/Danger/Pehl: "Interleaved Challenge Loop PUF: A Highly Side-Channel Protected Oscillator-Based PUF" in *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2022, vol. 69, no. 12, pp. 5121-5134 (©2022 IEEE) [21].