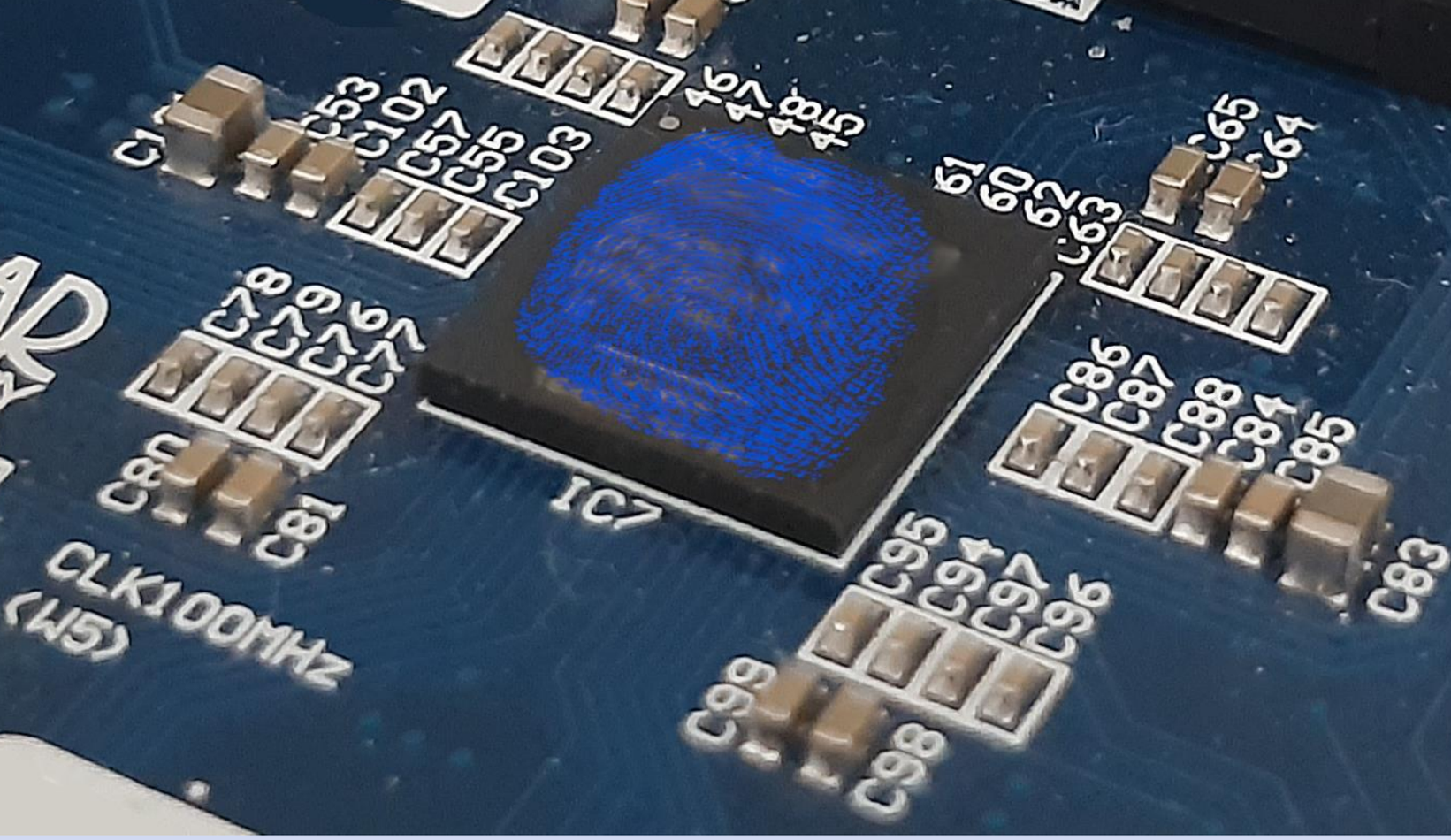


Design, Evaluation, and Application of Security Primitives that are Based on Hardware-Intrinsic Features

Michael Pehl

Habilitationsschrift

vorgelegt dem School Council der

TUM School of Computation, Information and Technology

Technische Universität München

Acknowledgement

This collection constitutes the end point of my habilitation process. Thus, it is the right moment to thank different people who encouraged me to start and continue this process and with whom I had the pleasure to work over this time. First of all, I want to thank my mentors, Georg Sigl, Jean-Luc Danger, and Ralf Brederlow, for the support they provided and their availability for several enriching discussions.

Modern research requires collaboration, and the different results presented in this work have required joint efforts, too. In this regard, I want to thank all the researchers and students with whom I worked over the last few years. In particular, I want to thank the previous and current doctoral candidates at the Chair of Security in Information Technology of the Technical University of Munich, especially the ones who worked or still work in my research group and were strongly involved in some of the research presented in this collection.

Carrying out research is a time-consuming and not always family-friendly endeavor. I, therefore, want to thank my wife and my two sons for their patience when I worked another time till late at night on my research. Without your presence and support, it would have been much harder to keep the motivation to handle the different challenges I have been facing in the last few years.

Munich, 29 February, 2024

Contents

On the Demand for Hardware-Intrinsic Security

1. Introduction

In our increasingly interconnected world, where small end-node devices significantly contribute to data collection and processing, these devices must be protected to ensure data confidentiality and privacy and to prevent IP theft and counterfeiting. Many devices in this context are out in the field and accessible to attackers. Therefore, in particular, hardware-related attacks must be considered. As a consequence, secret information needs to be stored and processed in a way so that attacks cannot annihilate security mechanisms. In other words, it is not sufficient to use algorithms that are mathematically secure. These algorithms must also be implemented in a way such that they do not leak information that can be revealed by observing channels correlated to the secret, so-called side channels. In addition, the system must remain secure, even if an attacker tempers with it. While the abovementioned holds for any hardware implementation of a cryptographic algorithm, this work focuses on mechanisms that exploit hardware-intrinsic security. But what is behind the term "hardware-intrinsic security"? The following gives a definition for this collection.

Definition 1.1 *Hardware-Intrinsic Security* refers to any kind of security mechanism that is based on physical features of the hardware not fully controlled by the manufacturer or any other person, including malicious parties.

The most apparent technology in this regard and the focus of this collection are Physical Unclonable Functions (PUFs): Every device suffers from process variations in the manufacturing step. For chips, e.g., fluctuations in doping concentration or oxide thickness can be named as such variations. Usually, such variations are minimized since they make the performance of a chip less predictable and can cause a large portion of produced chips to fail to meet specified performances. Nevertheless, variations cannot be canceled entirely and are not fully under the control of a manufacturer. In terms of hardware security, this means that devices have an intrinsic feature, which neither can be fully controlled nor can the concrete realization for a chip be predicted. Since this holds for the manufacturer and any other person, particularly for attackers, the features are candidates to become a root of trust. Silicon-based PUFs are on-chip measurement circuits that allow for deriving secret information from process variations. Their main applications are key storage, chip identification, chip authentication, fingerprinting, and tamper protection.

Remark: Through the concept of PUFs, hardware-intrinsic security is strongly related to the concept of physical layer security: Physical layer security normally refers to the concept of protecting secret information through the physical properties of the transmission channel between devices. Hardware-intrinsic security can be seen as a complement to this concept, which allows for deriving secret information from inside of a device. The concept of PUFs also relates hardware-intrinsic security to biometrics since the physical properties of a device can be seen as a kind of device fingerprint. As a consequence, several theoretical concepts from physical layer security as well as from biometrics are relevant and can be adapted to the domain of hardware-intrinsic security. However, the idea of hardware-intrinsic security as it is understood in this collection also goes beyond: It also covers implementations for on-chip security like probing detection or random number generation.

In the different works presented in this collection, mechanisms are researched, which contribute to the goal of making future hardware more secure by entangling security mechanisms with hardware features. More precisely, the contributions are mostly related to PUFs and consider:

- Methods to increase the security of hardware-intrinsic designs.
- Hardware-related attacks like Side-Channel Analysis (SCA) and Fault Injection Analysis (FIA), in the sense that security primitives are analyzed and improved in this regard. This includes new methods to protect against probing attacks and the development of new approaches to derive and process secret information.
- Statistical evaluation of PUFs as the primitives exploiting hardware-intrinsic features.

With these contributions, this collection significantly enhances and improves the state of the art.

2. Motivation and Summary of Contributions

The previous chapter has introduced on a high level what hardware-intrinsic features are. However, the questions of why they should be preferred in certain situations over classical security mechanisms and why it is worth advancing research in this domain remained unanswered.

To answer these questions, take the concept of PUFs as the most important example in this domain. PUFs have been investigated in scientific research for more than twenty years. During this time, they developed from a promising new idea to security primitives, which can be found nowadays in many products. They owe their success to the fact that they are based on hardware-intrinsic features and can be built in silicon. These features are present in any manufactured circuit and do not require additional steps in the fabrication. At the same time, it is deemed impossible to read out the relevant process variations of a chip from external. Thus, if the concrete realization of process variations on a chip is considered a secret, it can be regarded as, at first glance, more secure and, at the same time, cheaper than storing an artificial secret on the chip. This illustrates why hardware-intrinsic features are a relevant research field for security in general: If these features can be exploited, they might offer a high level of security at a low price since they do not rely on a specific technology and can be implemented with low effort. As such, they are also promising candidates that can be implemented in low-cost devices.

However, the research on PUFs has shown that it depends on their concrete implementation and application whether implementations based on hardware-intrinsic features are secure. Thus, the first bunch of contributions in this collection care about the implementation and application of PUFs. While key storage solutions based on hardware-intrinsic features exist and are already in the market, novel protocols and more efficient approaches are still of interest. In that research field, in particular, PUFs with challenge-response behavior are considered in this collection, and their application in a novel protocol has been researched. If such implementations are suggested, also the limitations must be well understood. For this purpose, the vulnerability of such approaches to machine learning is analyzed, showing a novel attack strategy pointing out the limits of storing secrets with PUFs with a challenge-response behavior. These results give interesting insights, e.g., regarding the error correction needed in this context to derive a reliable secret from noisy measurement data provided as PUF responses: They demonstrate that certain rather complex error correction codes might be

beneficial for such applications. Thus, a novel decoder for an error correction code – a decoder for a Polar code – was studied for its applicability in the PUF context. It was demonstrated that such a decoder has indeed a much higher latency and area footprint than classically used decoders but that it is still possible to implement such beneficial structures with reasonable effort.

As already mentioned, the security of hardware must take side channel leakage into consideration. For the context of hardware-intrinsic features, the claim is that reading a secret from off-chip is not possible. However, while this holds for direct measurement, it does *not* imply that it is impossible to observe the circuit that measures process variations on a chip or the processing of such measurement results into secret information. Thus, corresponding SCA attacks are of particular importance in the context of hardware-intrinsic features due to their exploitation in the context of lightweight devices that are virtually always accessible by an attacker. For the same reason, tampering with the device, which is done in order to insert faults and to conclude from observed faulty behavior on the secret, must be considered. In addition, hardware-intrinsic features can serve as detection mechanisms for attacks. The intrinsic runtime of a signal over a wire, e.g., can indicate a probing attack on this wire.

Since hardware-related attacks play such a significant role, the second bunch of contributions in this collection deal with it: The first part discusses the topic of side-channel attacks on a specific oscillator-based PUF. On the one hand, this provides insight regarding vulnerabilities of such PUF types given classical quantization techniques as well as novel, more efficient ways to derive secret PUF responses from PUFs. On the other hand, these contributions illustrate concepts of how to prevent attacks on PUF primitives. Two basic paths are taken: (i) As for classical cryptographic algorithms, randomness can be used to hide information from an attacker. Depending on the concrete strategy for quantization, the required randomness can be generated by the PUF on the fly or a significant number of random bits would be needed, resulting in the demand for an additional random number generator. This collection also discusses the potential implementation of a lightweight solution but shows that such an approach is doomed to fail. (ii) The second path suggests a method to overcome a particular attack by modifying the design so that the measurement of oscillation frequencies is no longer easily possible. Such a method allows for the protection of a PUF without the need for randomness.[1]

But not only research regarding the vulnerabilities of PUF primitives is presented in this collection. Also, the post-processing from a noisy PUF response to a reliable secret key was under research. In this regard, side-cannel attacks have already been considered in the past. The contribution in this collection takes another novel approach: It analyzes the susceptibility of a concrete error correction against FIA. It shows that clock glitching is a powerful attack on a PUF-based system: While the remaining entropy after an attack is higher than for a side-channel attack on the unprotected system, it can be mounted under less strict assumptions than SCA attacks on the same setting and might still be possible if masking countermeasures are implemented.

If secret information is present and transmitted on a chip, microprobing might be an attack vector. Therefore, in the second bunch of contributions, a novel probing detection mechanism is suggested and implemented, too. It exploits that the intrinsic delay of a wire is modified by attaching a microprobe, making the probe detectable. Effectively, similar strategies as for oscillator PUFs are exploited: By realizing an oscillation, the same wire is measured multiple times, improving the resolution of the measurement, and differential comparison of multiple oscillators is used to get rid of environmental effects. However, different from the PUF concept, any form of local process variations can become an issue for such a concept. Using a simple calibration technique, however, makes the concept suggested in this contribution practical.

The last bunch of contributions in this collection targets another critical issue in the design

[1] It shall be noted that yet unpublished research indicates that – while the attack seems indeed harder – novel mathematical concepts and the exploitation of other leakage points might be used to still mount an attack.

of security mechanisms that are based on hardware-intrinsic features: the evaluation. While first PUFs are in the market, they lack certification. I.e., chips with PUFs are certified but only in a context where PUFs do not provide the root secret. To overcome this issue, a consistent strategy for evaluation is needed, as it has been introduced, e.g., for True Random Number Generators (TRNGs). This collection provides contributions to this process in three regards: (i) It demonstrates a method for the evaluation of PUFs that respond to symbols from a higher-order (not only binary) alphabet. (ii) It discusses advanced entropy estimation methods for PUFs and for the key derived from a PUF. (iii) It demonstrates a strategy for a more systematic evaluation of noise effects when compared to what is typically used in literature. For these methods, the contributions of the collection also discuss how to take advantage of the evaluation results in a practical setting.

In sum, this collection contributes in different ways to the scientific and practical progress of the exploitation of hardware-intrinsic security. With a focus on PUFs as the dominating primitive to exploit intrinsic features, it contributes to the secure design, secure implementation, and security evaluation. The included contributions aim to close several gaps in research and to improve the understanding of potential weaknesses with the goal of enhancing the security and efficiency of methods exploiting hardware-intrinsic features.

3. Structure of this Collection

This collection is structured as follows: After the introduction into the field of hardware-intrinsic features in the first part, the second part provides the necessary background to understand the contributions of this collection and gives a survey of the state of the art. In this survey, important PUF primitives and background on how secrets are derived from hardware-intrinsic features are introduced in Chapter 4. With this background in place, Chapter 5 introduces the main applications of hardware-intrinsic features, namely (i) key storage, (ii) identification, authentication, and fingerprinting, and (iii) tamper protection. Chapter 6 provides an overview of how hardware-intrinsic security is evaluated today regarding statistical performance. Attacks on hardware-intrinsic security, as well as countermeasures for protection against such attacks, are outlined in Chapter 7.

The third part of this collection introduces different developments that contributed with the participation of the author to the further advancement of the state of the art. This part is structured in three chapters: Chapter 8 introduces findings that are related to general considerations in the design of key storage solutions. Chapter 9 discusses novel hardware-related attacks and countermeasures for PUFs and their post-processing and introduces a new probing attempt detector scheme. Lastly, Chapter 10 provides advancements in evaluation strategies for hardware-intrinsic features. These three sections provide each a summary of the findings regarding specific research questions, which are complemented by peer-reviewed publications that have been published through well-approved conferences and journals. After the demonstration of contributions, Chapter 11 concludes this collection and gives an outlook into the direction of future research questions.

Throughout this collection, publications with the contribution of the author are highlighted in bold in the text as well as in the bibliography. Bibliographic information regarding the publications that are part of this collection is provided at the end of the corresponding summaries in Part III. These publications which are part of this collection are also listed in the preamble of the bibliography.

II Background and State of the Art

4. Physical Unclonable Functions

4.1 Background

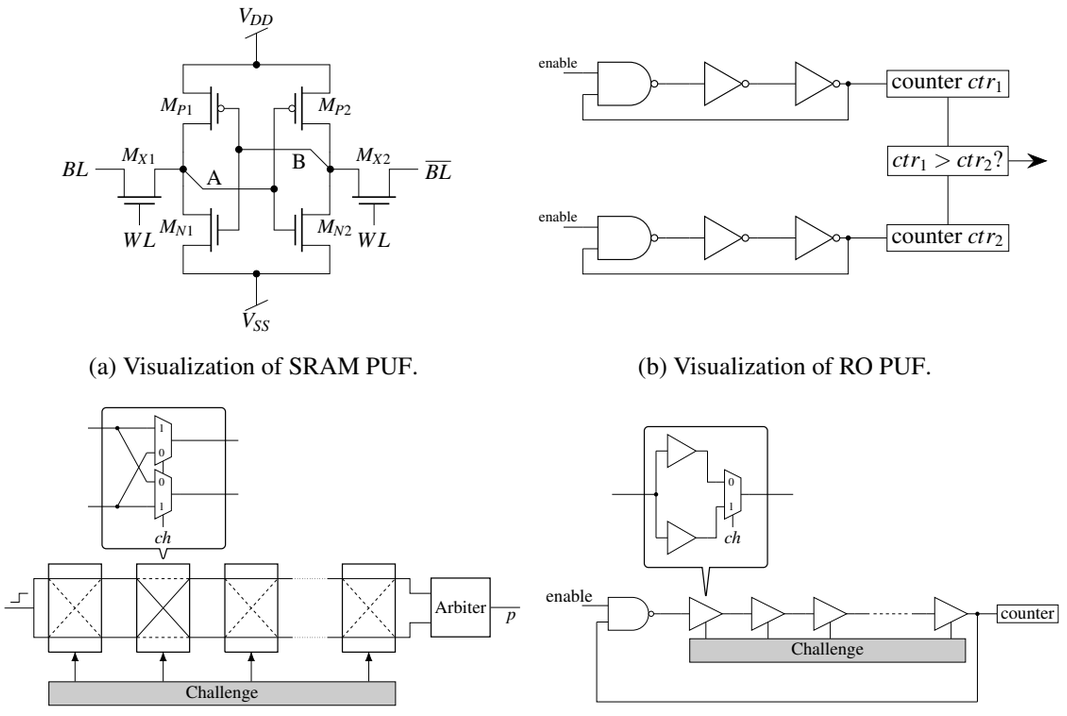
The focus within this work is on silicon-based PUFs, i.e., on PUFs integrated into chips. Unavoidable process variations from the silicon manufacturing step are exploited in this case to enable hardware-intrinsic security. These properties, like electron mobility or threshold voltage variations, typically cannot be observed directly. Thus, measurement circuits are needed, which measure a chip's performance related to the exploitable variations. These measurement circuits are mostly called Physical Unclonable Functions (PUFs), Physically Obfuscated Keys (POKs), or Physical One-Way Functions.

Definition 4.1 A PUF is defined in this collection as
- A *physical* measurement circuit, which is
- Mathematically and physically *unclonabel with reasonable effort*, and is
- Interpretable as a *function* mapping process variations and possible configuration challenges to some response.

Please note that the definition within this collection incorporates mathematical clonability and puts unclonability into a practical context of the use case. Furthermore, a PUF is a measurement circuit, and the measured characteristics are not perfectly stable over time but suffer from noise and aging and depend on operation conditions. Consequently, the *response* derived from a PUF, i.e., the quantized measurement result, is also unstable.

A wide variety of PUF circuits has been suggested to derive secrets from hardware-intrinsic features. The most popular are SRAM PUFs, Ring Oscillator PUFs (RO PUFs), and Arbiter PUFs, introduced below. In addition, the Loop PUF is presented, which was a research target in some of the works in this collection. This selection of PUF types is visualized in Fig. 4.1 and detailed in the following subsections.

PUFs have been categorized concerning different properties [94, 104]. One important distinction is their challenge-response behavior. The terms *weak PUF* and *strong PUF* are frequently used to describe the behavior in this regard. However, these terms are not uniquely used and imply strengths and weaknesses of a PUF, which might be misleading. Thus, the terms Single Challenge



(a) Visualization of SRAM PUF.

(b) Visualization of RO PUF.

(c) Visualization of Arbiter PUF.

(d) Visualization of Loop PUF.

Figure 4.1: Visualization of different basic PUF types.

PUF and Multi Challenge PUF are mostly used in this work.

Definition 4.2 Single Challenge PUF:
A Single Challenge PUF is a PUF that cannot be configured at runtime. I.e., neglecting noise, the same PUF response is generated whenever an enable signal is applied or directly at startup.

Definition 4.3 Multi Challenge PUF:
A Multi Challenge PUF is a PUF that can be configured at runtime with different challenges. I.e., the challenges modify the measurement circuit formed by the PUF.

Please note that within this collection, only digital challenges in the form of a bit vector are considered.

4.2 Silicon-Based PUFs

The currently dominating fraction of PUFs is based on silicon. This makes them easy to integrate and comparably cheap. In this section, some important representatives are described, which have high relevance in literature and research and are useful for the understanding of the remainder of this collection.

4.2.1 SRAM PUF

SRAMs have been introduced as PUF primitives in [64, 53]. They belong to the category of Single Challenge PUFs, which are commercially offered by Intrinsic ID[1] and are currently the most frequently used PUF type in practical applications.

[1] `www.intrinsic-id.com`

SRAM PUFs exploit the startup behavior of SRAM cells: If the SRAM cells are implemented as six transistor SRAM cells as depicted in Fig. 4.1a, all nodes are uncharged in the power-down state of the circuit. If the chip is switched on, a current flows from the supply V_{DD} through the PMOS transistors into the inner nodes, causing an increase of the voltage between nodes A/B and ground (V_{SS}). Due to process variations, the currents are different, so the voltages at nodes A and B increase at different speeds. The voltage of node A (B) is at the same time the gate voltage of the NMOS transistor below node B (A) and – when large enough – enables a current flow from node B (A) to ground. Due to the differences in the two paths, this results in a latching effect driving one of the two nodes A and B to logical one and the other one to logical zero. The secret provided by a single SRAM cell is *which* of the two nodes stores a logical one after power-up.

If the SRAM cell is perfectly symmetrically built, the state after power-up depends only on the process variations (assuming a noise-free case). However, the result is sensitive against the power-on ramp of V_{DD} [65] as well as on imbalance in layout and routing. Nevertheless, several researchers have shown that even the area-optimized SRAMs of Commodity-Of-The-Shelf (COTS) microcontrollers and other devices are suited to derive PUF responses with reasonable quality [162, 53, 64].

4.2.2 Ring Oscillator PUF

The Ring Oscillator PUF (RO PUF), with the basic building block shown in Fig. 4.1b, was introduced in 2007 [143]. It comprises as core elements two identically designed ring oscillators. Two competing ring oscillators are enabled for a fixed time, and the number of oscillations within this time is counted. Consequently, after this fixed time, the counters contain values proportional to the oscillation frequencies.

Although the ring oscillators are identically designed, they exhibit – even for a noise-free case – different frequencies since the elements in the rings all have slightly different delays due to process variations. A secret bit is derived by comparing these frequencies, respectively, the counter values; typically, the sign bit of the counter difference is taken as a secret.

While ring oscillators are Single Challenge PUFs like SRAM PUFs, they have the benefit that they are much easier to design: While the SRAM PUF must be designed in a symmetric way to provide good PUF quality, for the RO PUF, only the oscillators and their fan-out must be identical. This makes ring oscillators well-suited for FPGA implementations and, thus, a popular research target. Nevertheless, ring oscillators are hardly in favor of practical applications since they suffer from large area requirements (two oscillators are needed per bit), longer measurement time when compared to SRAM PUFs, and high energy consumption.

4.2.3 Arbiter PUF

The first Arbiter PUF was introduced in [51], where it was mainly used as a configurable delay chain in the configurable oscillator suggested as a PUF in [52, 51]. It consists of a chain of configurable delay elements, as depicted in Fig. 4.1c. Depending on a challenge bit, a stage either forwards the input to the output in the same order or crosses over the connection.

Several configurable stages are connected to derive a secret bit from the Arbiter PUF. A pulse is applied in parallel to both inputs of the first delay stage. Given a fixed configuration, the pulse propagates through two distinct paths from the inputs of the first delay element to the outputs of the last. Due to process variations in the delay elements, the pulse arrives at the outputs at different points in time. An arbiter – typically a latch – switches to either one or zero depending on which of the paths is faster, and the value is taken as a secret bit.

The Arbiter PUF as a Multi Challenge PUF, which takes an N_{ch}-bit challenge, can be configured in $2^{N_{ch}}$ ways. However, the paths are not fully disjunct, and the delay difference at the output can be interpreted as a linear recombination of $N_{ch}+1$ coefficients [88]. Thus, the response bits derived for

different challenges are not fully independent, which makes the Arbiter PUF vulnerable to machine learning attacks. Nevertheless, the Arbiter PUF is the base for most of today's Multi Challenge PUFs, such as the XOR Arbiter PUF [143], the Multiplexer PUF [134], or the Interpose PUF [115].

4.2.4 Loop PUF

The Loop PUF depicted in Fig. 4.1d was suggested in [25] as an easy-to-design PUF primitive and is part of the portfolio of Secure-IC Ltd.[2] It consists of a chain of configurable delay elements and an inverting gate in the feedback, forming a ring oscillator. A challenge bit selects one out of two possible paths with different buffers inside for each delay element. Since process variations influence the different buffers differently, switching a challenge bit is equivalent to changing the value of one delay element of the PUF.

The construction of the Loop PUF is, at first glance, similar to the configurable oscillator in [51]. However, the configuration mechanism of the delay chain is different from [51], making the Loop PUF much easier to design: A single wire connects the different delay stages for the Loop PUF. In contrast, like for the Arbiter PUF, two signals are transmitted over two wires in competing paths for the configurable oscillator. Consequently, the connections between buffers in the Loop PUF change the expected frequency but – with the bit derivation described below – do not bias the response independently from their routing. In contrast, for the configurable oscillator and, respectively, the Arbiter PUF symmetric routing of the interconnects of delay stages is crucial.

While the Loop PUF is a Multi Challenge PUF, it is usually used as a weak PUF for key storage. For this purpose, it exploits a specific method of challenge generation: N_{ch} challenges, each consisting of N_{ch} bits, are constructed using the concept of *Hadamard* codes. This code has the property that the Hamming distance between any pair of codewords is 50%. Consequently, the Hamming weight of all codewords neglecting the all-zero codeword is 50%, too. A bit is derived from the Loop PUF by applying a challenge **ch** and its bit-wise complement $\neg$**ch** and measuring the frequency difference between those configurations by counting the number of oscillations in a fixed time. Using **ch** and $\neg$**ch** means that all elements in the ring changed between the two measurements. Thus, the standard deviation of the difference in the frequency between **ch** and $\neg$**ch** is larger compared to only a few delay elements that would be changing. This can be expected to result in a higher proportion of large frequency differences, contributing to the Loop PUF's reliability.

Using Hadamard challenges with the described Loop PUF concept yields two further advantages: (i) Since the Hamming distance of all challenges is $N_{ch}/2$, the challenges are well spread over the challenge space, which contributes to the Loop PUF's unpredictability. (ii) Since the Hamming weight of all challenges and their complements, neglecting the all-zero/all-one challenges, is $N_{ch}/2$, for half of the delay elements, the upper, and for half of the delay elements, the lower path is selected. I.e., as long as all delay stages are equally designed, an internal imbalance in the placement and routing is compensated and does not cause a bias of the PUF response.

It shall be noted that the construction of the Hadamard challenges is easy to achieve: A Hadamard code is constructed in the later experiments of this collection by using the maximum length output sequence of an Linear Feedback Shift Register (LFSR) with a primitive polynomial and non-zero seed; I.e., from an l bit LFSR a $2^l - 1$ bit long sequence L is taken. L is padded in front with a zero. A cyclic shift of L retaining the zero from the padding constructs challenges that fulfill the properties of a Hadamard code at low hardware overhead. Since the Hadamard code scales with powers of two, the Loop PUF is typically defined with the corresponding length.

Given its straightforward design and good area utilization, the Loop PUF is a good PUF candidate. Its first main drawback is the comparably long measurement time: The Loop PUF is formed by a relatively long ring. For a sufficient Signal-to-Noise Ratio (SNR), i.e., a good

[2] `https://www.secure-ic.com`

distinguishability of process variation dependent oscillator frequency and measurement noise, mainly the number of oscillations is important so that a longer ring implies a longer acquisition time T. Furthermore, the $2 \cdot N_{ch}$ challenges are applied sequentially to the Loop PUF, resulting in an overall measurement time larger than $2 \cdot N_{ch} \cdot T$ for N_{ch} responses. The second drawback of the Loop PUF is that it is comparably easy to attack with SCA attacks. Hardening it against SCA attacks is one contribution of this collection.

4.3 PUFs in Emerging Technologies

Different new technologies have entered the domain of electronics in the last few years. The most promising ones to be mentioned are flexible and memristive devices.

Besides other application domains, flexible devices might be used in the future in a security-relevant context, e.g., for intelligent packages or electronic medical patches. This, however, causes additional challenges, one of them being the question of how to store a secret key on such devices. Using current approaches, it might be easy to read out the secret from a device [105]. The authors in [105], however, mention that using the PUF principle might be one solution to this issue but also faces additional challenges in this context, e.g., the fact that bending of the flexible circuit might influence the PUF responses. Overall, the security research for this kind of technology is in an early stage.

Compared to that, much more research has been done on the security of memristive devices. Memristive devices are, in the narrow sense, fundamental devices that relate charge and flux linkage as proposed by Leon Chua in [27]. Today, several technologies – e.g., Resistive RAM (RRAM) [141], Spin-Transfer Torque Magnetoresistive RAM (STT-MRAM) [68], Phase Change Memory (PCM) [17], and $BiFeO_3$ (BFO) memristors – have been and are still being developed, which behave as the proposed element.[3] They promise to be useful in several applications since they might be used as Non-Volatile Memory (NVM) or – at least some of them – programmable logic, fostering applications like in-memory computation and neuromorphic systems [129].

While some research regarding the implementation of cryptographic algorithms [9, 87, 90] in memristors and side-channel leakage [23] in these technologies exists, most security research has focused on the design of PUFs and TRNGs based on memristor devices. Memristor-based TRNGs, are frequently based on classical TRNG structures exploiting, e.g., the phase jitter of an oscillator [126]. In that, they try to improve the randomness by exploiting the specific random switching behavior of memristors. For instance, [77] generated random bits based on the stochastic delay time in switching.

Usually, the switching behavior of memristors is not only random in a temporal manner that can be exploited for TRNGs but also in a process variation-specific way that can be exploited for PUFs. Different methods of exploitation have been suggested, with the majority based on the comparison of the resistance of memristor cells [137] in a specific state. The resistance measurement includes strategies like the measurement of currents, in particular, the currents of sneak paths in a memristor array [127], by sophisticated direct measurement [20] or by differential comparison of the resistance of memristors [24].

An alternative approach towards PUF implementation with memristive technologies is to exploit their slightly different switching behavior. For magnetoresistive RAM, it has been suggested, e.g., to force them to take their preferred state and to use this information as a secret [31]. For RRAM it was suggested to let two memristors compete in a serial or parallel configuration so that – with high probability – only one is configured into the High Resistive State (HRS) or, respectively, Low Resistive State (LRS) and to take the information, which is configured, as a secret, e.g., **[8]**

[3] It is worth noting that the currently existing technologies might be no memristors in the narrow sense and that the proposed passive element might not exist [159, 2].

and [7]. A third way to construct a PUF from memristors is described in [130]: The write pulse to memristors in HRS is set to a length where not all cells are configured into LRS, but some remain at high resistance. Adapting the length of the pulse, the expected amount of HRS and LRS can be set to 50%. Which cells remain in HRS depends on process variations and forms the secret PUF response.

Overall, there are different approaches for using memristors as PUFs. However, there is no best strategy yet. It shall also be noted that the majority of the research for such emerging technologies has been done using simulation models or low-scale experiments. Thus, it can be expected that not all effects visible in actual implementations have been explored yet, and further research is required.

4.4 Quantization of Analog PUF Responses

While for some PUFs, like the SRAM PUFs, the response is inherently a binary value, it is derived for many PUFs from a digitized analog value, referred to as *analog response*, like a counter value or a voltage level. By using differential measurement or subtracting some reference value for de-biasing, the distribution of the responses is usually shifted to zero. The distribution of the accordingly processed PUF values is assumed for the rest of this section, and without loss of generality, as well approximated by a Gaussian distribution $\mathcal{N}(0,\sigma^2)$. This corresponds to the ideally approached distribution for most realistic cases. Five relevant quantization strategies can be identified for converting such digitized analog values into a secret response.

Sign-Based Quantization: The sign-based quantization takes the sign of a difference as the secret. In the assumed probability distribution, this corresponds to cutting the distribution into two equally probable halves and assigning one half the response bit zero and the other half the response bit one as depicted in Fig. 4.2a. While this is the simplest method to implement, it provides relatively low entropy per area compared to some quantization techniques below. In addition, most of the responses are under the practical assumption of approximately Gaussian-distributed analog responses comparably close to the decision bound and, thus, susceptible to disturbance.

Equiprobable Quantization: To increase the entropy per area, the sign-based quantization can be extended to an equiprobable quantization. This approach, which has been discussed initially in the context of biometrics [22], was discussed in the PUF domain, e.g., in [32, 73]. Fig. 4.2b depicts this approach. It can be seen that more equally likely symbols are derived from the same distribution. This increases the entropy derived per area with the PUF. However, due to the multitude of decision bounds, the sensitivity to disturbance is increased. Therefore, this kind of quantization is typically combined with a first stage of error correction: The (noise-free) analog response is shifted to the center of the interval it is inside. The information how much it is shifted is stored as helper data. Applying the same shift later to the noisy response increases reliability.

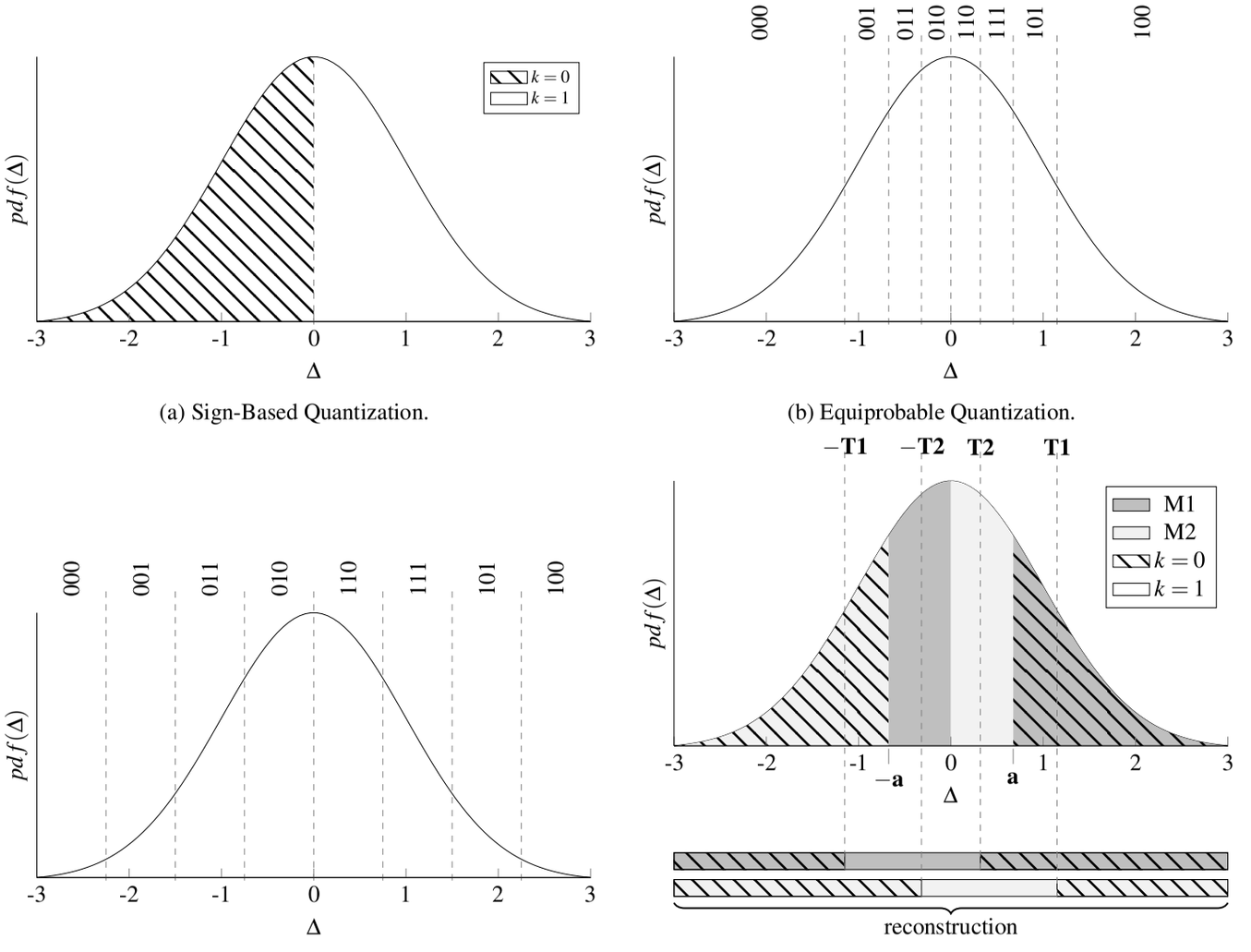
While the main benefit of equiprobable quantization is an increased entropy per area, it has not only the drawback of an increased noise level requiring a first error correction. It further requires the characterization of the probability distribution if the analog PUF response is not approximately normally distributed with a known mean and variance. Furthermore, it can be concluded from the findings in this collection that this kind of quantization makes a PUF more sensitive against side-channel attacks since not only the sign but also the magnitude of the analog PUF response is used.

Equidistant Quantization: Equidistant quantization, visualized in Fig. 4.2c, has been introduced to the PUF domain, e.g., in [73]. Similar to the equiprobable quantization, the goal is to derive multi-bit symbols from an analog PUF response; Different from the previous quantization, the intervals are of equal size but not equally probable. As a consequence, the symbols do not

have full entropy even if the PUF is ideal. Therefore, this strategy has been mainly suggested to reach a high tamper sensitivity even in the tail of the analog PUF response's distribution, where intervals in the equiprobable quantization are large and, thus, a significant change of the analog response by an attack has no effect when equiprobable quantization is used.
Since the derived symbols are not equiprobable, the response derived by equidistant quantization requires some form of security amplification, e.g., a hash function applied to the finally derived secret, compressing the secret with reduced entropy per bit to a secret with sufficiently high entropy. The findings of this collection allow, similar to the equiprobable quantization, the conclusion that this scheme can be potentially attacked via SCA attacks since, again, the magnitude of the analog PUF response is used. The work in this collection also suggests that the reduced entropy per symbol leads to an even higher decrease of entropy in a derived key.

Two Metric Helper Data Scheme: The Two Metric Helper Data Scheme (TMHD) was introduced in [30] to increase the robustness of a PUF. The scheme is depicted in Fig. 4.2d. The approach assumes a Gaussian distribution of PUF responses. A logical one or zero is derived according to the quartiles of this distribution; The inner two quartiles map to a logical zero, the outer two to a one or vice versa. Helper data is stored for each analog response, depending on the quartile it falls into during an enrollment phase. This helper data defines a metric for decoding: In Fig. 4.2d Metric $M1$ is stored during enrollment in quartiles two and four and $M2$ in quartiles one and three. For bit derivation from measurements in a so-called reconstruction, the stored metric is used. In this step, only a significant deviation of the analog PUF response from its reference value during roll-out can cause a wrongly derived response bit. The reason is the definition of the bounds for bit derivation for each metric according to octiles (bars in Fig. 4.2d), which are shifted compared to the enrollment stage. In addition, a re-characterization of the normal distribution's standard deviation for each reconstruction makes the approach more robust.
While the TMHD provides only one response bit like the sign-based quantization, it provides a significant reliability increase at the cost of the storage of helper data. The work in this collection has shown, however, that it introduces the same potential attack vector via the amplitude of the analog PUF response that has been mentioned for the previous quantization schemes. Another drawback of the TMHD is the re-characterization of the standard deviation during enrollment, which might lead to significant hardware overhead.

Lehmer-Gray Encoding: The Lehmer-Gray encoding has been used for PUFs in [96]. The approach sorts the analog PUF responses according to their values. The order represents the secret derived from the PUF. The secret order is computed and encoded using Lehmer-Gray encoding, which can be implemented efficiently even in hardware.
It can be concluded from [171] and [96] that deriving the secret from the sorting of the analog PUF responses is – neglecting noise effects – theoretically optimal with respect to extracted entropy. However, the mapping to a binary response causes that not all response sequences are equally likely, i.e., some bits are correlated or biased. This might lead to helper data leakage and causes that – like for the equidistant quantization – a compression to a full-entropy secret is required even for a perfect PUF. Also, the Lehmer-Gray Encoding is more strongly affected by SCA attacks than the simple sign-based bit derivation since an attacker can partly conclude from the amplitude to the ordering. However, while knowing the absolute value of the analog PUF response but not the sign breaks the TMHD completely and reduces the secret of an equidistant and equiprobable quantization to one bit – only two intervals can be reached with a known fixed absolute distance from the origin – a higher level of uncertainty remains in the Lehmer-Gray encoding.



(a) Sign-Based Quantization.

(b) Equiprobable Quantization.

(c) Equidistant Quantization.

(d) Two Metric Helper Data Scheme.

Figure 4.2: Visualization of different quantization schemes for PUFs. The symbols in Figures (b) and (c) are exemplarily defined using a three-bit Gray code. Figures (a), (b), and (d) are adapted from **[150]**.

5. Applications of Hardware-Intrinsic Features

Based on the primitives defined in Chapter 4, three applications of hardware-intrinsic features are introduced in this section. First, PUFs are discussed for the use case of deriving and storing a secret key. Second, the use of PUFs for authentication and identification is discussed, which is strongly connected to fingerprinting. Tamper protection is discussed in this section as a third application.

5.1 Key Derivation and Key Storage

One of the main applications for which hardware-intrinsic features are exploited is the storage of a secret key. For this purpose, on a conceptual level, a secret is derived with a PUF from hardware-intrinsic features, called *PUF-based key storage* in the following. If this secret is stable and has sufficiently high entropy, it can be used directly as a secret key. The benefits of such an approach, which are frequently mentioned, are:

1. *It is not possible to read out the key if the chip is off.* This claim assumes that it is not feasible to measure the used process parameters from off-chip. Under this assumption, the secret used as a key is measured on demand with a PUF. The secret is then stored in volatile memory or immediately used and removed from the system if no longer needed. Fulfilling this property ensures that – different from NVM – no permanently powered protection mechanisms like sensors are needed to protect key material in the power-off state of the system. In contrast, for different storage technologies, such as ROM, Fuses, Anti-Fuses, or Flash, methods for reading out the key in the power-off state have been shown or at least sketched, e.g., [85]. For emerging technologies like Ferroelectric RAM (FeRAM) or RRAM, research that shows the hardness of attacks on the memory is lacking. However, since these memories are non-volatile, it can be expected that a readout is feasible.
2. *The approach is technology-independent and low-cost.* Most PUFs are built from standard logic gates or at least with standard transistors. In parallel, all processes suffer from process variations. As a consequence, it is feasible to implement PUFs in any technology and for any system for which logic – and thus some cryptographic algorithm requiring a key – is used. In addition, PUFs that are easy to design and integrate are a low-cost but still secure solution. In contrast, some technologies, like Flash, are not available or too expensive in cutting-edge

technology nodes. Emerging storage technologies might be an alternative, too, but they are, until today, not sufficiently researched, as mentioned above.

3. *The approach simplifies the key-enrollment process.* Classical NVMs suffer from different problems regarding a key-enrollment process. ROM is not well suited for key storage since the storage content has to be defined via the masks in the manufacturing process. Since there is, in practice, only one chip design replicated multiple times on a waver and the masks are reused, this leads to the same key on all chips so that reading out a single key corrupts the security of all chips. Programmable NVM is primarily programmed from off-chip or requires some additional on-chip circuitry to provide e.g., high programming voltages. Since the goal is to embed a secret key, programming the key from off-chip must be done in a secure environment with a mechanism ensuring random keys per device. If the key is generated on-chip, in addition to circuitry programming the NVM, a good TRNG must be available on the chip.
   In contrast, when exploiting hardware-intrinsic features with a PUF, the secret is directly derived from hardware and – for a good PUF– inherently unique to a device. In practice, this benefit of hardware-intrinsic security is, however, frequently somewhat hypothetical: Also, for PUFs, specific circuitry for the enrollment process is needed, and the used secret key is frequently a random number, which is sometimes generated on-chip by a TRNG. In addition, so-called helper data need to be stored so that this benefit might be small.

While the use of a PUF-based key storage provides several benefits, the main challenge – besides constructing a high quality PUF in terms of unpredictability – is that the PUF responses are noisy. This inherently results from measuring process variations on demand, which suffers from measurement and quantization noise. In addition, environmental effects, like temperature and voltage variations, cause shifts in the analog properties and, therefore, changes in the binary PUF response. Also, aging causes an alteration of the exploited inherent process parameters, causing changes in the binary response. Since a key must be constant over time, this hinders the PUF response from being used as a key directly.

To solve the noise issue for PUFs, two complementary strategies are used: First, designers try to make the PUF as robust as possible. However, despite serious efforts to make the PUF perfectly reliable, today, no such PUF can be found.[1] Thus, the second approach, namely using error correction to correct a varying PUF response always to the same information, is needed. This strategy implies that a codeword must be available. Since an ideal PUF outputs a (fixed) random sequence, the output is usually no codeword. Therefore, two different phases are needed for storing a key with a PUF:

1. In an *enrollment phase*, a key is enrolled to the PUF. While the concrete implementation depends strongly on the Helper Data Algorithm (HDA) used, mostly some secret – either from the PUF or from a TRNG – is selected. Using this secret and a (possibly additional) PUF response, a HDA defines so-called helper data, which, together with a noisy version of the same PUF response, allow for the reliable derivation of a key. Different forms of HDAs are mentioned in Section 5.1.1, also providing a view on concrete realizations of the enrollment phase.
2. In a *reconstruction phase*, the enrolled key is reconstructed from a noisy PUF measurement. For this purpose, again, an HDA is used – now in most cases for mapping helper data and PUF response to a codeword. The codeword is then error corrected (cf. Section 5.1.2). The described process is depicted in Fig. 5.1: An analog PUF response $\mathbf{p}_{raw}$ is derived from a PUF primitive and quantized to a possibly noisy response. This response is mapped with helper data to a codeword, which is corrected to a secret. In practical applications, the secret

[1] Some publications claim a perfectly reliable PUF. However, these PUFs normally turn out to be some form of the previously mentioned NVM or are strongly biased and thus insensitive against noise and environmental conditions.

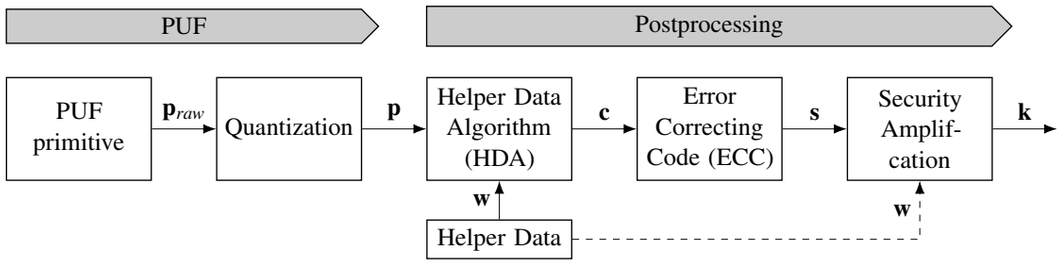


Figure 5.1: Schematic representation of a PUF key-derivation.

is frequently further processed in a so-called security amplification step. The reason is that, in case the PUF does not have full entropy, the helper data can leak about the secret, and – for some HDAs as discussed in Section 5.1.1 – an attacker can manipulate the helper data to reveal the secret key. The security amplification step is typically implemented as a hash function, compressing the secret after error correction optionally together with the helper data to a key with sufficient entropy per bit. While the boundary is not sharp, the measurement with a PUF primitive together with quantization is commonly referred to as "the PUF" and the steps from the digital PUF response over HDA, Error Correcting Code (ECC), and security amplifier to the secret key are usually referred to as *postprocessing*.

The process using helper data in order to enable error correction and to derive a secret key is also used in biometrics to extract a secret from noisy data. A generalized form of it has been formalized in [40]. In the PUF context, mostly two parts are distinguished: HDA and an error correction part. The resulting conjunctions are known as either *Secure Sketches* or *Fuzzy Extractors*.

Before discussing HDAs and error correction in more detail, consider how the key is used in a PUF-based key storage solution. In very general, the PUF is reasonably used to store a private key for asymmetric crypto **[29]** or a symmetric key. The question is how this key enters the device. Since in the common PUF scenario no secure NVM is available, it is hard to ensure that an attacker cannot re-trigger an enrollment process. Suppose the attacker is able to select the key, which is embedded during enrollment. In that case, they learn from this key and the helper data immediately the PUF response, and forward and backward security is affected. Thus, the key must generated on the device, either by a TRNG or from the PUF directly. For asymmetric crypto, this solves the problem since the private key can stay on the device, and only a public key must be output. Using the in this way generated PUF-derived key, however, directly as a pre-shared key, would break the security of the device since it means that this key must be output during enrollment. The attacker can trigger a new enrollment process and learn from the output key and the new helper data the PUF and from the PUF and the previous helper data the previous key. As a consequence, the most secure use case of PUFs in a key storage scenario is to use them as a key-encryption key **[76]**. I.e., the key is generated on the device and is used to encrypt – and on-demand decrypt – some otherwise unprotected memory, in which certificates or pre-shared and private keys are stored. If an attacker now re-triggers the enrollment process, they can observe only the change in the helper data. No attack is known until today which can exploit this change of the helper data.

5.1.1 Helper Data Algorithms

After the introduction of the key storage scenario in the previous section, this section introduces the first stage of postprocessing, namely the Helper Data Algorithm (HDA). These algorithms are used to construct a codeword from random data. Two categories are most commonly used in the PUF context, which are named in this collection as *code-based* and *bit-selection* or *pointer-based* approaches.

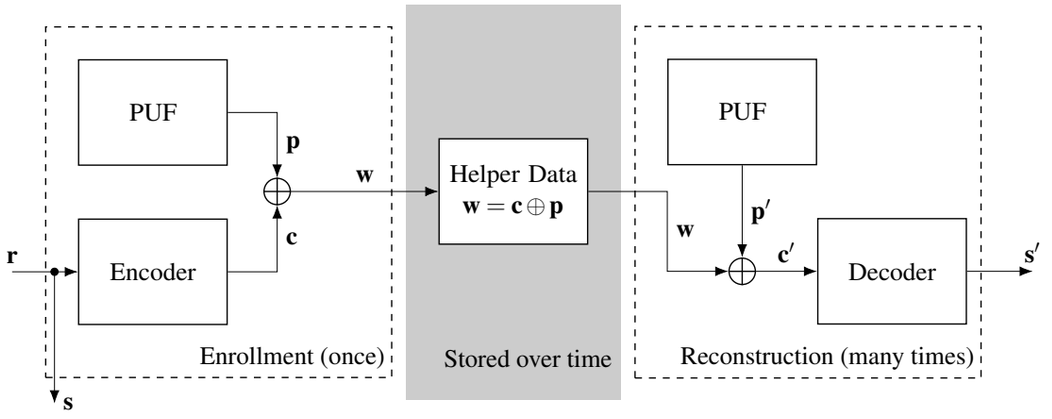


Figure 5.2: Schematic representation of enrollment and reconstruction with the Fuzzy Commitment scheme.

Code-Based Helper Data Algorithms

The algorithms in this class have in common that secret data are encoded or decoded with an error correction code in the process of helper data generation. All of these algorithms use PUF responses $\mathbf{p}$ to generate helper data $\mathbf{w}$; some require on top some random seed $\mathbf{r}$, which stems either from a random number generator or from a PUF. For the same PUF, the different HDAs in this group result in a different amount of helper data and different leakage properties as summarized, e.g., in **[122]**. Four important helper data algorithms in the PUF context are introduced in the following to highlight the basic working principles and the differences between the schemes:

Fuzzy Commitment: Since several parts in this collection take the Fuzzy Commitment scheme as an example, this scheme is explained in more detail than the schemes below. The scheme was already introduced in [78] and is visualized with its enrollment and reconstruction phase in Fig. 5.2.

In this scheme, the secret $\mathbf{s}$ is chosen during enrollment at random, i.e., $\mathbf{s} = \mathbf{r}$. In the enrollment phase, $\mathbf{s}$ is encoded to a codeword $\mathbf{c} = ENC(\mathbf{s})$ and helper data is generated by XORing the codeword with a PUF response

$$\mathbf{w} = \mathbf{c} \oplus \mathbf{p}.$$

This can be seen as masking the secret codeword with the PUF. It has been shown, e.g., in **[122]**, that for a perfect – in terms of predictability, i.e., unbiased and uncorrelated, yet noisy – PUF, the helper data do not leak about the secret for this scheme. Nevertheless, the helper data leak about the PUF, a circumstance exploited in the contributions in Section 8.2. To reconstruct the secret from the helper data and a now noisy PUF response $\mathbf{p}'$, the process from the enrollment phase is reverted: The helper data is XORed with $\mathbf{p}'$, which yields the noisy codeword $\mathbf{c}' = \mathbf{p}' \oplus \mathbf{w}$. For a reasonable selection of the error correction code, the decoding of this codeword $DEC(\mathbf{c}')$ results in a reconstructed secret $\mathbf{s}'$ that is with high probability the same as $\mathbf{s}$.

Code-Offset Construction: The code-offset fuzzy extractor was introduced in [40] as an adaption of the Fuzzy Commitment scheme to their systematic and is, thus, equivalent. Only the perspective on the data changes. Again, in the enrollment step, a random number $\mathbf{r}$ is encoded to a codeword $\mathbf{c}$; Again, the codeword is XORed with the PUF response $\mathbf{p}$ to receive the helper data $\mathbf{w}$. However, now $\mathbf{p}$ is considered the secret. This can be seen as a secret $\mathbf{p}$ is masked with an imperfect mask since the codeword contains redundancy and thus does not have full entropy for an attacker knowing the code.

To reconstruct $\mathbf{p}$, the noisy codeword is computed as $\mathbf{c}' = \mathbf{p}' \oplus \mathbf{w}$ and processed in the case of a correct decoding to $\mathbf{c} = DEC(\mathbf{c}')$. The error pattern identified by a decoder in this step might be used directly to reconstruct the PUF; alternatively, $\mathbf{p} = \mathbf{c} \oplus \mathbf{w}$ can be used.
While this scheme leaks about the PUF and thus the secret, which is reconstructed, the entropy for an attacker knowing the helper data is the same as in the Fuzzy Commitment case. A compression of PUF response is necssary but also sufficient to receive a secret, which can be used for further cryptographic applications.

Systematic Low Leakage Coding: This scheme, introduced in **[63]**, is merely the same as the Fuzzy Commitment scheme. However, the random number now comes directly from the PUF, and a systematic encoding scheme is used. As a benefit, no helper data needs to be stored for the information part of the codeword since the noisy PUF response can be used for this part of the codeword in the reconstruction. The redundancy part is generated by encoding the information part; helper data is generated by XORing this redundancy part with additional PUF response bits. Since only helper data for the redundancy part are stored, this approach reduces the amount of helper data by the information length.

Syndrome Construction: This approach was introduced again by [40]. Different from Code-Offset Construction and Fuzzy Commitment, it does not require additional randomness. Instead, it takes a reference PUF response and computes its syndrome for a given code. This syndrome is stored as helper data. Like in the other schemes, the helper data carries in this way information about the – assumed to be perfect – PUF, but not about the secret. To reconstruct the correct PUF response, the noisy PUF response is decoded, and the minimal modification of the noisy PUF is searched, which results in the syndrome stored as helper data. This process sounds complicated at first glance, but for many codes, the decoder can be modified to decode to the syndrome stored as helper data rather than to the – usually for the error-free case required – syndrome $\mathbf{0}$.

Bit-Selection Helper Data Algorithms:

The general idea of bit-selection helper data algorithms is to select from a large number of PUF bits responses with sufficient reliability. Thus, these approaches provide a first stage of error correction. However, the reliability of the response after this step is usually not sufficiently high, so that still a subsequent error correction stage is needed. For this purpose and depending on the approach, either responses that are not only sufficiently reliable but also form a codeword for subsequent error correction are selected, or the chosen responses form a random but more stable bit stream, and one of the code-based helper data algorithms is applied on top.

Bit-Selection helper data algorithms do not leak information about the secret information[2]. However, since stable bits are selected, they leak reliability information of the PUF. This is typically not critical for Single Challenge PUFs. However, for Multi Challenge PUFs, reliability information can enable machine learning (cf. Section 7.3). Moreover, if an attacker can manipulate helper data used to select bits, it is possible to compare bits and retrieve a secret in this way (cf. Section 7.1). In the PUF literature, mainly three types of bit-selection HDAs are of relevance:

Dark-Bit Masking: Dark bit masking introduced for PUFs, e.g., in [6, 103, 136], is the simplest form of bit-selection helper data algorithm. For every bit, one helper data bit is needed, storing the information if the corresponding PUF response is sufficiently reliable or not. Despite the potential vulnerability through helper data manipulation attacks, the main drawback of this approach is the relatively large number of helper data required. This is also the reason why the two subsequent schemes have been suggested.

[2]When the selected bits form a codeword, they leak about the PUF, since they reveal for a given code how bits depend on each other just like the code-based helper data algorithms.

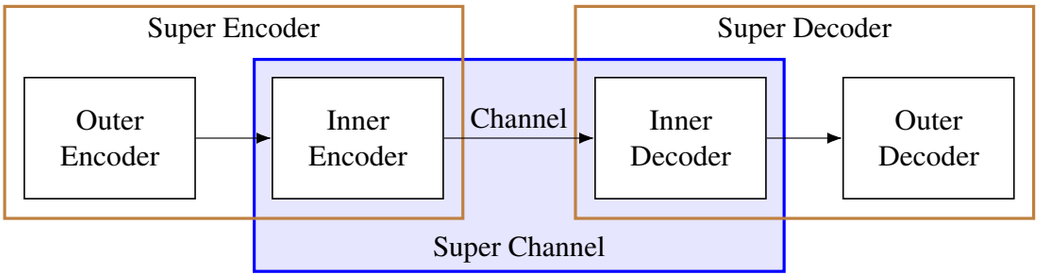


Figure 5.3: Schematic representation of a serial code concatenation: The concatenation consists of an outer and inner code. The corresponding encoders and decoders can be seen as super encoders and super decoders. Alternatively, the channel, together with the inner encoder and decoder, can be seen as a super channel with improved noise characteristics.

Differential Sequence Coding: Differential Sequence Coding (DSC), introduced in [61, 62], can be seen as dark-bit masking, in which reliable bits are stored using a prefix code. For this purpose, the position of each reliable bit is stored in the form of a pointer indicating the distance from the previous reliable bit. A variable length prefix code encodes this difference in the sequence of bits. The main benefit of DSC is a significant reduction of helper data. However, this number is only probabilistically bounded due to the use of a variable length code.

Index-Based Syndrome Coding: Index-Based Syndrome Coding (IBS) [172] considers the responses of a PUF a sequence and splits it into equally sized blocks. To store a codeword, it stores in its basic form one bit of the codeword per block of PUF responses. For each of the responses, it determines the probability of being a logical 1, which is the mean error probability for bits that are expected to be logical 0 and the reliability for bits that are expected to be logical 1. To store the codeword bit, it generates the index of the bit that has the highest probability of being equal to the codeword bit – e.g., the one with the lowest one-probability for a logical zero – as helper data. Note that the idea is to select the most reliable bit. But if no bit with the correct expectation is present in the block of PUF responses, the least stable bit with the wrong expectation is taken, and the error probability can go beyond 50% for a single bit. Sufficiently long blocks reduce the probability of such a degenerated case and make the approach practical. Furthermore, instead of storing a codeword bit directly, selecting the most stable PUF bit per block independently of its value and combining IBS with one of the coding-based HDAs solves this issue.
Another approach, which promises an improved error correction capability, is to use Complementary Index-Based Syndrome Coding (C-IBS) [60]. Here, dependent on the codeword bit a pattern 0101... or 1010... is embedded into a block of PUF responses. The process is similar to IBS, just that the indexes of the most stable ones and most stable zeros within a block are stored in decreasing order of reliability.

5.1.2 Error Correction Schemes

As it has been mentioned in Section 5.1, Fuzzy Extractors and Secure Sketches in the PUF context normally consist of an HDA and an error correction code. An overview of such codes is provided, e.g., in [59, 58]. To summarize, most error correction schemes used in the PUF context are based on linear block codes. To handle the large error probability of a PUF at the input – usually a bit error rate up to 15% or even 25% as in [75] is assumed – and to stay with an acceptable hardware overhead, most commonly code concatenations [43] are used with a serial concatenation being the most widely used one in the PUF context (cf. Fig. 5.3).

In the encoding of such a serial code concatenation, the information is first encoded by an outer code. The resulting codewords are then symbol-wise further encoded by an inner code. After transmission, the inner decoder first reconstructs the codewords for the outer decoder and corrects in this step parts of the errors. The outer decoder corrects further errors and reduces the error probability in the information according to its specification down to the required level (in the PUF context, frequently a key error probability of 10^{-6} or 10^{-9} such that an error in the key is less likely than a breakdown of the complete device). Outer and inner encoder and, respectively, decoder together can be seen as super encoder and decoder.

In code concatenation, the role of the inner decoder is to reduce the error probability from a very high rate down to a level that the outer code can efficiently handle. The search space for possible concatenations is vast, and the quality of existing constructions is not discussed in this work. Nevertheless, as a design rule, the inner code must have a very low rate [39] since such codes are well suited to bring down very high error probabilities to an acceptable level for other codes. Typical choices in the PUF context are repetition codes or small Reed-Muller codes. However, these codes are not efficient in achieving low block error (and thus key error) probabilities and are, therefore, not suited as outer codes. For these outer codes, most commonly Bose–Chaudhuri–Hocquenghem (BCH) codes are used in the PUF domain since they have a low area footprint in hardware, resulting in low cost, which is one primary goal in many key storage solutions with PUFs.

Besides the mainstream decoders, few works have considered soft decision decoding for PUFs [157], stand-alone codes like LDPC-codes [144], or use error correction codes implemented in software [79]. Also, the contribution in Section 8.3 uses a stand-alone Polar Code with soft decision decoding. In general, soft decision decoding bears a great advantage since soft information, i.e., reliability information about PUF bits, increases the error correction capabilities and thus decreases the amount of required PUF and helper data bits. Since the usage of PUF bits as well as of helper data bits shall be minimized to reduce costs, using soft information seems to be a reasonable choice. However, the decoders for such codes usually are much more complex. Hence, the area for more complex decoder hardware mostly outweighs the benefits from reducing PUF bits and helper data. Furthermore, soft information is only available on the fly for certain PUFs, where digitized analog responses are available. Storing soft information increases the helper data size significantly and bears the risk of enabling reliability-based attacks only on the base of the helper data, although no such attacks are currently known for Single Challenge PUFs.

Using software implementations for error correction is only reasonable if the code is implemented in ROM or write-once-read-many memory or if other means prevent the manipulation of the code: The error correction code requires direct access to the PUF. If such software can be modified and run without any restriction of permissions, e.g., on a microcontroller, the attacker can modify the code and output the secret. If the PUF provides, e.g., the master key, there is no means to ensure the integrity and authenticity of the error correction in software and to hinder such an attack. Therefore, a hardware implementation of the error correction seems crucial or at least beneficial in many PUF applications.

5.2 Identification, Authentication, and Fingerprinting

The use of physical properties for authentication in modern technology is commonly attributed to Bauder [10]. However, in that work, optical effects were used. A first suggestion for identifying electronic chips using hardware-intrinsic properties was provided in [91]. They suggested a circuit[3] that measured process variations of different transistors. Since this first work indeed only tried to *identify* chips, it was sufficient to ensure a unique response of the circuit per chip. They were,

[3]The term PUF was not introduced at that time but the concept was the same. Thus, the term PUF is used for this concept, too.

therefore, able to output the PUF responses directly since the identifier is, per definition, not secret. They also did not have to care about the learnability of the PUF or replay of the response since these are no threats in the identification scenario.

In contrast, most applications today focus on device (not data) *authentication* with PUFs. When PUFs are used for authentication, which was first suggested in scientific publication for optical PUFs in [117, 118], the requirements are much stronger: Now, a verifier must be able to verify the authenticity of the prover based on a PUF's response. To realize this goal, prediction, as well as replay of the response of a prover's PUF for a specific authentication round, must be prevented. The concept used for PUFs is, thus, similar to a challenge-response protocol with classical crypto.

Recall that for simple challenge-response authentication with symmetric crypto, a pre-shared key is used. The verifier sends a randomly chosen challenge that is encrypted by the prover and sent back. The verifier can check that the prover used the correct key, which ensures authenticity. For PUFs, there are two options to realize such a protocol. The straightforward way is to keep the protocol with a classical cryptographic algorithm and to store a pre-shared key with the PUF as discussed in Section 5.1. This would retain the security properties of the protocol – given the PUF key storage is secure – but requires the overhead of the classical cryptographic algorithm as well as of the postprocessing of the PUF response to an error-free key. To be more lightweight, another path is usually taken: A Multi Challenge PUF is used in the protocol. For each authentication round, a unique challenge is applied to the PUF, and the response is used to authenticate a device. Since the PUF response is noisy, errors can occur in the response, which is typically accepted in these PUF-based challenge-response protocols. In a way, the hardware-intrinsic features are seen in such a protocol as a key, the challenge remains the input, and the measurement circuit, which is configured by the challenge, becomes a kind of a keyed hash function.[4] Such a protocol is secure under the assumption that the PUF's challenge-response behavior is indeed unpredictable independent of how many challenge-response pairs have been seen by an attacker and that the attacker cannot read out a large portion of the challenge-response space and pre-compute this way future responses. Unfortunately, today, there is no confidence that this security requirement is met by any PUF construction (cf. Section 7.3). Therefore, besides making PUF constructions more and more complex, protocols are used to overcome security issues (cf. Section 5.2.1).

Related to identification and authentication is the use of PUFs for device *fingerprinting*. Fingerprinting is frequently required for the unique identification of a chip over its lifetime or within the production chain of complex systems. To clarify the meaning with respect to security requirements, two cases are of high relevance:

1. The chip is delivered for assembly or replacement, and a fingerprint shall be used to show that it is genuine. This use case boils down to an authentication scenario based on, e.g., a PUF-based challenge-response protocol, with all the strong requirements for authentication included. The reason is that the only feature used in fingerprinting today is the PUF response itself, and the protocol on top must ensure that the PUF response is not generated by a fake device.[5]
2. The second use case is that a complete system can be brought into a trusted state where all components are known to be genuine. In this situation, a combination of identification and tamper protection (cf. Section 5.3) suffices to prevent late unauthorized replacement of components: The tamper protection needs to ensure that the verifier of the fingerprint communicates with the correct device by showing that the communication channel is unaltered and the device has not been replaced. The transmitted fingerprint shows the device's iden-

[4] Please note that the first term used for PUFs was physical one-way function, which describes this concept quite well.

[5] As an analogy from biometrics, consider providing a human fingerprint without seeing the person who delivers it, which is obviously subject to faking. Similarly, it is not sufficient to trust in a PUF response without knowing that the correct chip – e.g., in a package – gives the response.

tity. Both together can be seen as an alternative way of authentication, providing sufficient confidence in the integrity and trustworthiness of a system for many use cases.

Scenarios for fingerprinting are currently under research, e.g., in **[89]**, since supply-chain and lifecycle protection play an increasing role. However, today, no conclusive solution exists which sufficiently solves the task of fingerprinting for all scenarios.

5.2.1 PUF-Based Protocols

Since PUF-based protocols are not only important for PUF-based authentication but might gain importance in the light of fingerprinting, this subsection summarizes some important findings and sketches a selection of three authentication protocols. In addition, some ideas for key distribution protocols are shown. [16] gives a comprehensive overview of a large variety of abstract protocols that go even beyond what has been done in practice until today but under the assumption of a perfectly unpredictable PUF. In contrast, the three protocols, which are sketched in the following, aim to make machine learning harder for practical Multi Challenge PUFs that suffer from a certain predictability. The protocols are selected for this collection since they require no significant hardware overhead like cryptographic hash functions, error correction encoder, or decoder.[6] If such postprocessing of the PUF response is needed, a crypto-based challenge-response protocol, possibly in combination with a PUF as a key store, is likely the better choice.

Remark on Generic PUF-Based Challenge-Response Protocols

What all protocols considered in this section need is an onboard TRNG. This is already needed in the most basic PUF protocol to prevent leakage of reliability information, which can enable or simplify machine learning [12]: An attacker who has complete control over the challenge can apply the same challenge multiple times, observe the response, and estimate the response's reliability in this way.

The most common strategy to prevent such attacks is to use a *challenge-sharing* approach. I.e., PUF device, as well as verifier, contribute half of the challenge, and the two challenge parts are concatenated. An attacker having control over a half challenge has – depending on the challenge length – a negligibly small probability of observing the same challenge-response pair twice. Please note that this causes a paradoxical situation: In order to use the challenge-sharing concept and to protect the PUF, the verifier needs to be able to predict the response of the PUF not only for pre-enrolled challenges but also for any arbitrary one. Thus, the verifier requires a model from the PUF. But the goal for a Multi Challenge PUF is to construct it in a way that is hard to model by an attacker. The standard method suggested to overcome this problem is to use a Multi Challenge PUF like the interpose PUF, which is composed of easy-to-model Multi Challenge PUFs (e.g., Arbiter PUFs). During an enrollment step, access is granted to the easy-to-model PUFs so that a model of the PUF can be built efficiently, and the access is disabled afterward – e.g., by blowing a fuse – so that modelbuilding for an attacker is much more complicated.

Selected Protocols to Make Machine Learning Harder

Even if a challenge-sharing strategy is used and a complicated PUF is used, the number of challenge-response pairs that can be used before the PUF device gets predictable is frequently too low. Thus, advancements on the protocol level have been suggested. While in this section only three selected protocols are considered, a more comprehensive study is given with further protocols analyzed, e.g., in [33].

Slender PUF Protocol: This protocol was suggested in [99]. For a challenge seed (generated with challenge-sharing) and a required authentication tag length z, a sequence of $Z \gg z$ challenges is deterministically generated, and the corresponding responses from the PUF are read. The

[6] Please note that – in addition to the required hardware overhead – error correction in a challenge-response protocol bears the problem that a huge amount of helper data must be transmitted or stored.

PUF device replays, however, not the complete sequence Z, but only a subsequence of length z, where the start index i is randomly chosen between 0 and $Z-1$ and bits i to $i+z$ (all numbers modulo Z) are output. While this protocol makes machine learning harder, it does not cause a larger number of challenge-response pairs to be required. Rather, it requires the attacker to train more models with the goal that an attack becomes practically infeasible.

Noise Bifurcation Protocol: For this protocol, introduced in [174], again, $Z \gg z$ bits are generated by a PUF, where z is the targeted number of bits used for authentication. To prevent the attacker from learning the PUF, the response string is split into Z/b blocks of size b. In each block, one bit is randomly selected. The selected bit is always transmitted; all other bits are dropped. However, only if all bits in the block are equal the transmitted bit is used for authentication: A verifier or attacker receiving a bit-stream knows which block a transmitted bit belongs to; they do not know which challenge-response pair in the block was selected for transmission. A verifier knowing the expected challenge-response pairs in the sequence of length Z, e.g., from a model, can, however, drop all transmitted responses for blocks with unequal bits. Responses from blocks with all bits expected to be equal – i.e., from blocks where it does not matter which challenge-response pair was transmitted – are taken for authentication.

Since the attacker has no such selection process, they can train either an exponentially large number of models or accept an increased level of noise. In the latter case, the attacker guesses for each block which challenge was selected; If the guess is correct or – by random chance – the response of the guessed challenge is the same as the correct one, they gain information for the model; otherwise, they add a wrong challenge-response pair to the training data, increasing the noise.

Compared to the Slender PUF protocol, the noise bifurcation protocol tries to make machine learning more complex so that more challenge-response pairs are needed. It comes, however, with two downsides: (i) The number of bits used for authentication is only probabilistic. I.e., Z can be chosen so that on average z blocks contain only ones or zeros. However, for some runs of the protocol, there are more and for some less such blocks (and therefore bit to authenticate with) available, depending on the concrete response sequence. In addition, (ii) the choice of the block length b is very limited. While a larger b seems beneficial from a security perspective since it increases the chance that the attacker sees a block where they might make a wrong guess, the probability of receiving a block with b equal bits decreases – for a good PUF – exponentially with increasing b. Since the number of expected blocks with equal bits must be kept at z at the same time, the required length of the response sequence Z increases exponentially and becomes quickly impracticably large.

Lockdown Protocol: The third protocol introduced in this selection, which was suggested in [173], takes a different approach. Instead of trying to make machine learning harder, it accepts that machine learning is possible and prevents it by restricting the number of used challenge-response pairs so that an attacker has too few challenge-response pairs for training. To understand the concept, note that for machine learning, the attacker requires a sufficient number of challenge-response pairs. This is easily possible for most PUF devices since the commonly used protocols ensure only the authenticity of PUF devices for the verifier. No authenticity of the verifier is guaranteed. In addition, devices using PUF-based authentication normally have no means to store a counter value securely and to track in this way the number of authentications already processed.[7] As a consequence, the PUF device responds to any query and reveals a potentially unlimited number of challenge-response pairs to an attacker. The lockdown protocol hinders this: The prover (PUF device) and the verifier (server)

[7] This is because there is normally no sufficiently secure memory, such as embedded Flash, to store a sufficiently large counter value in lightweight devices.

agree on two challenge seeds using challenge-sharing. The verifier uses a PUF model and sends a response sequence to the device, which is computed from the model for challenges deterministically derived from one combined challenge seed. Suppose the transmitted response sequence for the commonly agreed challenge seed is sufficiently similar to the noisy response sequence measured on the device for the same challenges. In that case, the device is unlocked (the verifier has shown its authenticity to the prover). The PUF device now measures the PUF and generates another response sequence using the second challenge seed; afterward, it sends the second response sequence to the verifier. The verifier compares the received sequence with the model and can verify the authenticity of the prover. This way, the two parties are *mutually authenticated*. Since the verifier in PUF protocols is typically a server in a trusted environment, which needs to protect already the PUF model, it can also keep track of the number of used challenge-response pairs and can limit it to an amount that is not sufficient for machine learning.

While the device cannot be authenticated any longer when its maximum number of challenge-response pairs is used up, this protocol effectively hinders machine-learning attacks. An attacker cannot query responses from the device since they would first need to know the response of the device for a new – partially by the device chosen – challenge seed. If they query responses from the verifier, counting the authentication rounds at the verifier side hinders reading out a sufficient number of challenge-response pairs for machine learning. Thus, the best the attacker can do is to eavesdrop on the communication between the prover and the verifier. Still, the number of challenge-response pairs that can be observed is limited.

The discussed protocols are useful in the light of authentication with a PUF and allow for a sufficiently good PUF to run lightweight challenge-response protocols without the overhead of classical cryptographic algorithms. In particular, in combination with modern PUF constructions requiring several millions of challenge-response pairs for learning, this might be a sufficient solution in many cases. However, there is no security proof ensuring that such a lightweight construction is secure in the medium or long term.

Key Distribution and Key Agreement Protocols with PUFs

Another branch in PUF-based protocols suggests storing keys with the help of Multi Challenge PUFs, as listed in [33] or described in [156]. Generally, these approaches are pretty similar to challenge-response authentication: A Trusted Third Party (TTP) knows the PUF and can either build a model or do other characterization steps for later key enrollment. For key agreements between two parties, the TTP selects for all device challenges and/or transmits helper data such that the derived keys are identical. To achieve this goal, a recent suggestion was to search for related challenge-response pairs [19] so that different challenges can be used for different devices and no helper data is needed. While this would provide a high level of security, this approach does not account for noise in PUF responses, which may limit the practical use. To account for noise, the TTP can generate additional helper data so that every device derives the noise-free response. It is also not necessary in this case to derive the same response on all devices so that all devices decode to the same key after error correction since a corresponding offset from the devices-specific keys can be transmitted with the helper data **[120]**.[8]

Two inherent problems appear with the mentioned protocols. First, since a symmetric key is shared, it is critical to use the PUF response directly as a key: Suppose one device is corrupted, and the channel to the other devices is eavesdropped. In that case, an attacker receives challenges from the channel and responses from the corrupted device and can use this information for machine learning. A cryptographic hash, hashing the PUF response to a key, can be used to prevent such

[8] Please note that compared to [19], **[120]** was practically implemented but needs to store more sensitive information at the TTP.

an issue **[120]**. Second, as soon as helper data are used, the helper data leak about the PUF (cf. Section 5.1.1). This collection shows in Section 8.2 the impact on the usability of such approaches in practice.

PUF-based protocols for key distribution, in which a specific key shall be embedded from a server to a device, are pretty similar to protocols for key agreement of two or more parties. From an attack perspective, the main difference is that the attack vector over a second device receiving the same symmetric key vanishes. However, since this attack vector likely has no significant impact in practice since a hash can prevent exploitation, and the attack vector via helper data remains, the two cases can be considered similar from a security perspective.

5.3 Tamper Protection

Another field where hardware-intrinsic properties play a crucial role is tamper protection. In this context, measurement circuits are built, which are sensitive against manipulation.

Tamper-Resistant Envelop

The first direction, which is strongly related to PUFs, is a tamper-resistant envelope [71, 72]. Such envelopes consist of a material that contains electronically measurable parts. In the referenced case, it consists of a foil in which wires are embedded. The wires are arranged as a mesh in two layers, forming capacitances at the crossings. Similar to PUFs integrated into chips, manufacturing-dependent variations, as well as bending the foil, cause differences in capacitances between wires. The measurement of the capacitances can be used in two ways: (i) Since manipulation of the envelope causes distortion of the foil or even destruction of certain capacitances, tampering attempts can be detected. (ii) From the foil-specific variations, a device-specific key can be derived. Attempts to tamper with the device cause not only an alarm but also destroy the key derived from the foil.

Since tamper detection with a secure envelope is related to PUFs, the questions are relatively similar. Two specific points should be mentioned: In order to detect tampering, other – more precisely equidistant – quantization schemes might be preferable [73], and since the measurement is noisy, once again, error correction is needed, but this has to be adapted to the slightly different use case [50].

Probing Detection

Related to the measurement of on-chip properties is also the implementation of on-chip probing detectors [100, 160, 161]. Probing detectors are an alternative to passive countermeasures against invasive attacks that try to probe, e.g., a bus structure. The basic idea usually is to detect the capacitive load that is added to a wire by attaching a probe. Probing detectors are related to the PUF concept in the sense that they also measure on-chip properties and that they suffer from noise. However, different from PUFs, for probing detectors, process variations are typically rather disturbing, and a calibration step and careful design are needed to get rid of such effects **[57]**. The work in Section 9.3 suggests one possible solution to this problem, which is related to oscillator-based PUFs.

6. Evaluation of Hardware-Intrinsic Security

When talking about evaluating the quality of hardware-intrinsic features, two primitives are of high importance: TRNGs and PUFs. Classical design metrics, like area footprint, timing properties, or energy consumption, play, however, a secondary role for those. More important is the unpredictability of the output, i.e., of the random sequence provided by a TRNG or the responses provided by a PUF. In the case of PUFs, the noise behavior, i.e., the number of faulty bits in a PUF response, is an important criterion, too. While, at first glance, for both primitives randomness is important, they have fundamental differences regarding testing **[5, 4]**: First, a single TRNG generates a virtually arbitrarily long sequence of data. At the same time, for PUFs, the same primitive under the same configuration is expected to always output the same response. Second, PUF reliability suffers from noise and environmental conditions. In contrast, TRNGs do not suffer from noise, but environmental changes might influence their predictability. The mentioned differences can be attributed to the different nature of the randomness exploited by the two primitives. TRNGs utilize run-time noise (typically thermal noise) while PUFs use variabilities due to the manufacturing process. The consequence is that a single TRNG can be tested for its quality, but a PUF needs to be tested always in conjunction with other realizations.

Since both TRNGs and PUFs exploit randomness, statistical testing is appropriate to judge their quality. However, for PUFs, the requirement of multiple instances of the same PUF frequently leads to low significance in tests. Furthermore, for PUFs, a higher dimensional problem is given and must be tested as detailed in Section 6.2.

6.1 A Short Summary of True Random Number Generator Testing

TRNGs are expected to output random sequences that are neither correlated nor biased.[1] For this purpose, noise effects like the phase jitter of oscillators [84, 119, 42] or direct thermal noise [66] are exploited. Since a monolithic TRNG circuit generates a long stream of random numbers, two major test strategies exist for testing TRNGs.

[1] In addition, the sequence must be forward- and backward-secure, i.e., an attacker knowing parts of the random sequence cannot predict future and, respectively, past random bits. This is, however, natively given if the sequence is derived from noise effects and has no statistical defects.

On the one hand, there are *entropy tests* to evaluate the TRNG quality. Given a long sequence of independent and identically distributed (iid) symbols, entropy estimation is easy to achieve, e.g., by counting the number of zeros and ones in a binary sequence and applying the appropriate formula for entropy. Testing entropy for non-iid sequences is more complicated and requires the application of bounds. While a good compression algorithm can compute an upper bound,[2] in some test suits for TRNGs, like in the one described by the NIST SP 800-90B [155], estimates for the minimum entropy of a sequence are suggested. The latter approach has the benefit that the estimates are expected to be more pessimistic, strengthening security claims.

On the other hand, there are *statistical tests*. These tests are generally used to qualify if the output of an TRNG can be considered iid unbiased. For this purpose, the tests compare properties of an observed sequence with the expectation for an iid unbiased sequence. They check, e.g., if the number of occurrences of zeros and ones in a sequence is as expected, if a specific pattern occurs repeatedly, or if any periodicity can be identified in the sequence. Such tests are suggested not only as a first step to the entropy test in the NIST SP 800-90B but also form the core of several other test suites like the one described in the NIST SP 800-22 [133], the German BSI's AIS 31 [81, 82], the Diehard and Dieharder test suites, and several more.

While the statistical tests above provide confidence that the symbols in a sequence appear iid, they do not ensure that the sequence is truly random. Therefore, e.g., in AIS 31, an additional model is required, which provides confidence that the observed randomness indeed stems from an unpredictable and non-reproducible random physical effect.

6.2 Backround on PUF Testing

Different from TRNG testing, several dimensions must be considered to evaluate the PUF quality. One reason is that different ways to guess a PUF must be distinguished:

1. *An attacker knowing the PUF responses for a given configuration on N_x chips, can try to guess the response on the $N_x + 1$st chip.* This scenario reflects the case that an attacker buys several chips, reads out the PUF, e.g., by a destructive attack, and tries to mount an attack on a yet unseen chip. To prevent such an attack, the tests must ensure that responses from any collection of N_x chips do not leak information. This implies the necessity to test the responses for each configuration and for each PUF position on a chip over all chips. In addition, it implies that the attacker shall not find a statistical weakness inherent to all chips, leading to the next test dimension.
2. *An attacker knowing the responses of PUFs at N_p positions on a chip shall not be able to predict the response for the $N_p + 1$st position.* This requirement is equivalent to demanding no statistical weaknesses like bias or correlation on a chip. Please note that practical PUFs frequently violate this requirement, and postprocessing, e.g., hashing, is needed to make the overall PUF-based construction secure. Of course, the same must hold over challenges, giving another test dimension.
3. *An attacker knowing the response of a PUF at a specific position and on a specific chip under N_{ch} different configurations (challenges) shall not be able to predict the response for $N_{ch} + 1$st challenge.* Again, this requirement is violated by today's Multi Challenge PUFs, which can be easily concluded from their vulnerability against machine learning.

Due to the different ways of guessing a PUF, it is not sufficient to test one dimension, but all possible dimensions must be tested. This is visualized as a response cuboid in Fig. 6.1 initially suggested in [67] and later in **[123]**. The columns in this cuboid correspond to the PUF response generated by a circuit at the same position measured on many chips, the rows correspond to the

[2] This is related to Shannon's source coding theorem, from which it follows that compressing a sequence to its minimum length gives an estimate for its entropy.

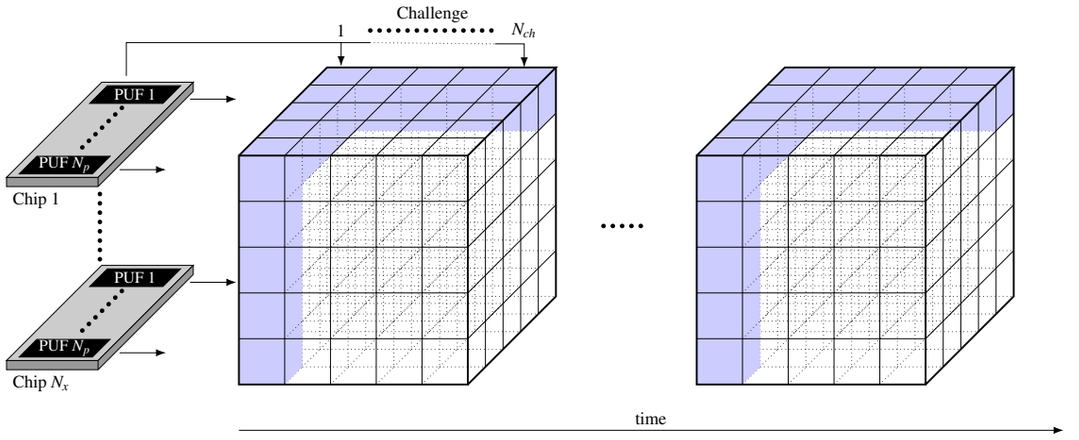


Figure 6.1: Visualization of dimensionality of PUF evaluation. Columns correspond to PUF response generated by a circuit at the same position measured on many chips under the same challenges; rows correspond to responses of the same PUF under different challenges; lanes in the z-axis correspond to PUF responses generated from circuits on different positions on the same chip under the same configuration. The development over time corresponds to noise and environmental effects.

responses of the same PUF under different challenges, and the lanes in the z-axis correspond to PUF responses generated from circuits on different positions on the same chip.

It is important to note that the order in which the different testing dimensions, and even different spatial dimensions on a chip, are recombined is crucial for the test result **[123]**. If random number tests are applied to a concatenation of PUF data, they can easily provide wrong results. Thus, testing the different dimensions separately is favorable. The test dimensions bear different challenges in this regard: A test over chips implies that multiple test chips with access to the PUF responses must be available. Thus, end-of-line testing of this property is not possible without introducing a potential backdoor for an attacker[3] and testing this property is expensive since it requires the manufacturing of a significant number of test chips. Tests over positions on a chip must partially consider their spatial relation and suffer from a limited number of positions available on a chip, leading to a limited significance of tests. Analyzing the PUF responses over challenges usually does not have the problem of too little data; However, an exhaustive test is usually not possible due to the exponentially large number of challenges, and the test result significantly depends on the selection and order of challenges. As a consequence, for the PUF quality evaluation over challenges, normally machine-learning is suggested (cf. [47] and Section 7.3), which is not further discussed in this section. For the remaining two test dimensions, dedicated PUF tests have been and are still being developed.

Before giving an overview of test metrics available for PUFs, the last dimension of testing in Fig. 6.1 should be mentioned: The PUF response alters over time due to random noise, aging, and environmental conditions. This effect usually does not cause additional predictability issues. However, it causes errors in the PUF response. Consequently, in case the PUF response is used directly, like in a challenge-response protocol, errors must be accepted, and the required prediction accuracy for an attacker is reduced. In case the PUF response is used as a key, a higher error rate results in a more complex error correction scheme. This does not only mean that the complexity and cost of the system are increased; In addition, more information is leaked about the system via

[3]For production testing, access to the PUF could be allowed, e.g., via a fuse, which is destroyed after the test. However, such a fuse might be reenabled with the help of some Focused Ion Beam (FIB).

helper data and potentially also about the secret if there is any imperfection in the PUF.

6.3 State-of-the-Art Metrics for PUF Unpredictability and Reliability

After the first introduction into the background in the previous section, this section aims to give a brief overview of the essential PUF metrics for unpredictability and reliability. An exhaustive discussion of most metrics can be found in [163]. The most common metrics have been suggested by Hori et al. [67] and Maiti et al. [97] with an overview and comparison of both given in [98]. The tests focus on binary PUF responses and are mainly based on the Hamming Distance (HD) and Hamming Weight (HW) of such responses.[4] The original tests by Maiti et al. were dedicated to a Single Challenge PUF scenario with an RO PUF. In contrast, the tests by Hori et al. focused on a Multi Challenge PUF scenario with Arbiter PUFs. To analyze a Multi Challenge PUF scenario with HW- and HD-based metrics, [67] breaks down the long response sequence of the Arbiter PUF into IDs. Each ID corresponds to a number of responses to pre-selected challenges, which has two consequences: (i) The scenario is closely related to the test of Single Challenge PUFs. Comparing, e.g., the ID for the same challenge sequence among chips is equivalent to comparing a response sequence of multiple Single Challenge PUF among chips. (ii) The result of the test is only meaningful for the selected challenge sequence and partitioning into IPs; choosing different challenges and/or other partitioning can result in a different PUF quality prediction. Given this remark, several metrics in [67] and [97] are very similar and mainly differ in their normalization.

It should be noted that [67] also discusses the confidence of evaluation results, which is valuable from a statistical point of view. However, these findings are not further developed or used later. The discussions below, however, also involve metrics from others, and in these metrics, hypothesis testing and confidence intervals frequently play a role.

6.3.1 Evaluation of PUF Reliability

Before discussing the PUF unpredictability, a short review of the state of the art regarding PUF reliability assessment is provided. In the PUF context, this is usually defined using the mean *intra-chip HD*: Given some expected binary outcome $\mathbf{p}_{ref}$ for a sequence of PUF responses $\mathbf{p}$ on a single chip, an error in the PUF response is defined as a bit-flip in $\mathbf{p}$ compared to $\mathbf{p}_{ref}$. Thus, the number of errors in $\mathbf{p}$ is $\mathrm{HD}(\mathbf{p}, \mathbf{p}_{ref})$. Averaging the HD from multiple measurements of $\mathbf{p}$ for the same chip – and possibly over multiple IDs as in [67] – results in the mean intra-chip HD and normalizing the result to the length of the $\mathbf{p}$ results in the fractional mean intra-chip HD of a chip. The latter can also be seen as related to the chip's Bit Error Rate (BER). The *Reliability* [97] or *Corrrectness* [67] of a chip is defined as

$$1 - \text{fractional mean intra-chip HD}.$$

The aim of having a single reliability value is to compare not only chips but also PUF structures and to design appropriate error correction codes. Nevertheless, only the average reliability of chips is usually reported. [67] suggested on top a metric called *Steadiness*, which provides a min-entropy value per chip to evaluate the reliability; However, this metric is usually not used in literature and skipped here for the sake of brevity.

It is worth noting that the concrete reliability value also depends on the reference point $\mathbf{p}_{ref}$. There are two reasonable choices for this reliability value: It can be defined as (i) an arbitrarily chosen readout of the PUF or (ii) the rounded mean (or median) of multiple readouts. In the first approach, the error probability of a single response bit is above 50% if the reference response deviates at some position from its expectation, which is likely to happen for several hundred-bit

[4] Given binary vectors $x = [x_0, ..., x_N]^T$ and $y = [y_0, ..., y_N]^T$, $\mathrm{HW}(x) = \sum_{i=0}^{N} x_i$ and $\mathrm{HD}(x,y) = \sum_{i=0}^{N} x_i \oplus y_i$.

long sequences and error probabilities per bit in the percent range. The second approach better approximates the expectation so that the likelihood of an error probability beyond 50% converges with increasing sample number to zero. Which of the two approaches is appropriate for a specific PUF depends on the application scenario. Suppose the estimate is used to design error correction or to define the error-tolerance level of some challenge-response protocol. In that case, the choice should reflect the enrollment process of the application.

Beyond binary data, there is currently no commonly agreed definition of reliability for PUFs. The reason is that for higher-order alphabets, the effect of a symbol error depends on the later-used error correction. If correction is done on a symbol level, it might be, for instance, irrelevant to which value a single symbol changes but only if it is faulty. In contrast, if a symbol is mapped back into the binary domain and error correction is done there, the change in the binary representation of the string is relevant but, at the same time, depends on the chosen binary encoding of the symbols.

6.3.2 Evaluation of PUF Bias

When evaluating the unpredictability of PUFs, usually the noise-free PUF response is considered, which is approximated by taking the most frequent response from a set of noisy measurements. Thus, if not stated differently, no time-variant behavior is considered for these metrics.

The first effect, which must be analyzed to prevent an attacker from having an advantage in guessing the PUF using statistical properties, is bias. Bias can appear along different axes, e.g., if the arbiter in an Arbiter PUF is not symmetric, bias will appear over challenges; If, e.g., the capacity of a wire close by the Ring Oscillator (RO) of a RO PUF causes a specific RO having a lower frequency, a position of the RO PUF over chips might be biased. The mean along different axes of the response cuboid in Fig. 6.1 is used to analyze bias effects in a PUF. This leads to the following metrics in the PUF literature:

Uniformity is defined in [97] and approximates the mean along lanes in the response cuboid. I.e., the relative frequency of ones (the normalized HW) in the response is taken for every chip without considering multiple challenges. This results in a distribution of relative frequencies over all chips. If the Uniformity of a chip differs from 50%, an attacker has an advantage in guessing the chip's response starting with the corresponding expectation. This is particularly relevant if all chips are biased and the mean Uniformity differs from 50%.

Randomness is defined in [67] based on the relative frequency of ones in all IDs generated on a chip. The min-entropy formula is applied to the resulting estimated one-probability to get the Randomness. This is closely related to Uniformity, and the attacker has an advantage in guessing if it deviates from its optimal value. However, while Uniformity operates on the dimension of positions, Randomness operates on the dimension of challenges, i.e., rows in the response cuboid. Therefore, this metric is also relevant for challenge-response protocols, where an attacker can observe a certain amount of responses, and an offset in the randomness might be computable by an attacker on the fly to predict yet unseen responses.

Bit-Alias is defined in [97] and is computed as the relative frequency of ones per PUF position on a chip with no challenges considered. This corresponds to averaging over a column of the response cuboid. As for Uniformity, an attacker knowing these statistics has an advantage in guessing when starting with the most likely response; However, since Bit-Alias approximates the one-probability per position, the optimal guessing strategy must be computed based on these probabilities. Ideally, the one-probability for every position, and thus the Bit-Alias, is 50%. If a bit on a specific chip position is considered chosen from a Bernoulli experiment and due to the limited sample size, however, the expected number of ones per position over multiple chips follows a binomial distribution. Consequently, the Bit-Alias can be seen as chosen from a normalized form of a binomial distribution.

Confidence Interval and Hypothesis Testing for Bias Tests. Since the previous tests are based on observations of a statistical distribution **[167]** suggested for Bit-Alias hypothesis testing and the computation of confidence intervals, which can be transferred to the other bias metric under the assumption of uncorrelated data. Regarding confidence intervals, the general conclusion is that the best PUFs with one probability at 0.5 are the hardest to test and that large sample sizes with observations of several hundred PUF bits are needed to provide sufficiently confident results. Regarding hypothesis testing, **[167]** suggests under the assumption of independent observations to formulate the alternative hypothesis stating that the analyzed PUF has too low quality so that denying this hypothesis results – with high confidence – in a statement that the PUF's Bit-Alias is within predefined bounds. Please note that due to the statistical nature of the experiment, it is expected that a certain amount of experiments (e.g., Bit-Aliases for several positions) fail the tests, which must be considered when interpreting the results of this, but also of the previous tests.

From the metrics mentioned in this section, Uniformity, and Bit-Alias are mostly reported for PUF primitives. However, only the mean values of these metrics are usually provided, and no confidence intervals are reported.

Remark: Most metrics focus on specific dimensions in the response cuboid and operate on reliability or unpredictability. Only a few metrics mix these different evaluation goals from which one should be mentioned here for completeness:

Proability of Misidentification: This metric, introduced in [142], predicts how likely a chip is to be identified as the wrong one. For this purpose, two probabilities are needed: the likelihood that a specific HD of PUF responses between chips appears and the likelihood that noise causes an error leading to misclassification for precisely this distance. These probabilities for misclassification under a specific distance are then weighted with the likelihood that this distance is observed and summed up. While the question targeted by this metric is important, the metric has the drawback that for a closed formula, it assumes iid PUF responses and iid noise among PUF responses, which might not hold in real-world applications.

6.3.3 Evaluation of Higher Statistical Moments

Besides bias in the PUF response, statistical defects of a PUF observable in higher moments are a concern. In particular, tests to cover correlations are available. It seems obvious to determine a correlation coefficient between PUF responses. However, not all dimensions are suited for this. In particular, the challenges applied to a PUF or the order of chips is usually not natively given.[5] Thus, only correlations of bits on a chip are considered. Two suggestions are mentioned here:

Spatial Autocorrelation was introduced into the PUF domain in **[165]**. Three approaches were suggested: Moran's I, Geary's c, and Join Count statistics. While Moran's I and Geary's c can be used for discrete and continuous data and are therefore suited if analog data are available, the Join Count statistics is dedicated to discrete data. Concept-wise, the approaches are relatively similar: A reference statistic for the case of an uncorrelated PUF is computed. The observed PUF response on a chip is compared to this statistic. For the Join Count statistics, e.g., the number of neighboring ones and zeros within a particular spatial distance is counted and compared to a reference. From this comparison, a z-score can be computed. If the z-score for a specific chip falls into certain limits, the hypothesis of correlated responses cannot be denied. Confidence that no systematic correlation effect appears in a design is gained if the distribution of the z-scores for all chips fits the expected normal distribution.

Fisher-Yates Test was suggested for PUFs in **[125]** and is another hypothesis test based on the

[5] The ordering of chips might be determined, e.g., from the positioning on a wafer, which would add another dimension to testing and require information that is typically unavailable to an attacker. Therefore, this option is neglected.

Null-hypothesis H_0 of independent PUF responses. It considers two neighboring[6] binary PUF responses $(\mathbf{p}_1, \mathbf{p}_2)$, and the occurrence of the events $(0,0),(0,1),(1,0)$, and $(1,1)$ is counted. The result is compared to a reference statistic for the uncorrelated case. A p-value is computed, defining the likelihood of an event as extreme as the observed one under H_0. Comparing this against a threshold α allows for denying H_0 if the p-value is too small. As for Spatial Autocorrelations, the expected outcome of all comparisons given a correlation-free PUF includes certain cases where H_0 is still denied (type I error). Confidence in a correlation-free PUF can be gained if the overall distribution of p-values fits the expectation.

While the two approaches are statistically sound methods providing good results even for small data sizes, two other suggestions have been made earlier in the literature: Uniqueness and Diffuseness.

Uniqeness by [97, 67] is likely today's most commonly used metric to evaluate PUFs. It is based on the so-called inter-chip HD – i.e., the HD between response sequences from different chips – without considering challenges in [97] and averaged over the challenge dimension in [67]. Regarding the response cuboid, it is related to the HD between all lanes. The Uniqueness is computed from these HDs by averaging all possible recombinations of responses dropping symmetrical pairs. I.e., , for three response sequences $\mathbf{p}^{(1)}, \mathbf{p}^{(2)}, \mathbf{p}^{(3)}$, the HDs $\mathrm{HD}(\mathbf{p}^{(1)}, \mathbf{p}^{(2)}), \mathrm{HD}(\mathbf{p}^{(1)}, \mathbf{p}^{(3)}), \mathrm{HD}(\mathbf{p}^{(2)}, \mathbf{p}^{(3)})$ are computed, summed up, and normalized, but the symmentric HDs $\mathrm{HD}(\mathbf{p}^{(2)}, \mathbf{p}^{(1)}), \mathrm{HD}(\mathbf{p}^{(3)}, \mathbf{p}^{(1)}), \mathrm{HD}(\mathbf{p}^{(3)}, \mathbf{p}^{(2)})$ are not considered. The idea of Uniqueness was originally to estimate how well the different responses can be distinguished. This implies that this metric should regard not only bias but also correlation effects. However, counter-examples can be easily found for extreme correlations like for a correlation of -1 between all responses, and the information provided by this metric is essentially the same as for the Bit-Alias metric above **[121]**. Nevertheless, it is worth noting that Uniqueness has been generalized in [74] for Higher-Order Alphabet (HOA) PUF responses.

Diffuseness was introduced by [67] to analyze the difference between different IDs on the same chip. Mathematical, it is closely related to Uniqueness, but the HD is taken between IDs instead of between chips, and it is computed for each chip separately. Nevertheless, the same limitations as for Uniqueness hold, and the metric cannot be reliably used to detect higher statistical moments.

6.3.4 Entropy Estimation for PUFs and PUF Derived Keys

Assuming that any bias or correlation in the raw PUF response might be acceptable if appropriate postprocessing is applied, the crucial question is if the derived secret has sufficient entropy. In particular, there are two points where measuring entropy is relevant: First, the entropy at the PUF output is relevant as a quality attribute of the PUF. Second, the entropy in a PUF-derived secret after error correction is relevant since adequate compression is needed in order to derive a key fulfilling specific entropy requirements.

Entropy Before Post-Processing

While some of the previously mentioned metrics like Randomness in [67] apply a min-entropy operation – this is applying $-\log_2(P, 1-P)$ to some probability P – the result is only a min-entropy for these metrics if the iid assumption holds for the data from which P is estimated. In practice, this assumption is frequently violated since the bias at different positions differs or correlations between response bits exist. If the entropy of a PUF with such realistic effects should be measured, joint entropy can be computed. It is, however, not feasible to compute joint entropy in the PUF context exactly, so an estimate is needed.

For PUFs, there are merely two attempts at estimating the joint entropy: Today, some publications try to estimate the entropy using the entropy tests described in the NIST SP 800-90B [155].

[6] In principal arbitrary positions can be compared, but correlations are most likely between neighboring bits.

However, it has been shown that the result of entropy tests significantly depends on how the PUFs' multidimensional data are serialized for the test **[123, 121]**. Furthermore, [139] suggests that the test results from the test suite vary widely and might overestimate the actual entropy, although they aim to estimate a lower bound for entropy.

The more mature approach to estimating the entropy of a PUF is to apply lossless compression in order to receive an upper bound for entropy. An upper bound for entropy cannot be used to qualify the output of a PUF as sufficiently good since it only states what the entropy of the PUF is at most. Still, it can serve as an indicator for the PUF quality if it is tight enough and can be used to disqualify a PUF for a specific use case. The challenge in this context is to compute a tight upper bound with only a few data given. Context Tree Weighting (CTW) [168, 169] was suggested in [69] for estimating the entropy of optical PUFs and is the most widely used approach to estimate an upper bound for the entropy of any PUF, since that.

Entropy After Post-Processing

Even more critical than the entropy before any postprocessing is the entropy in, e.g., the secret key. After error correction, a cryptographic hash function can always be applied to compress a longer secret with insufficient entropy per bit into a key with sufficiently high entropy per bit. However, it is essential to know the entropy of the post-processed PUF response for such a step in order to ensure a sufficiently long secret and a sufficient compression rate. This available entropy is not only influenced by the PUF but also by the leakage through helper data.

Given the statistics of a PUF, [35] suggested a method to estimate the conditional min-entropy in the secret after error correction conditioned to the helper data known by an attacker. The approach is generally applicable; however, in this early form, it was only feasible to compute the entropy for PUFs, which have either iid biased responses or the same correlation between responses. To improve the practical applicability, **[164]** suggested an extension, which – accepting some error in the estimate – allows to consider also PUFs that have different chip-position-dependent bias. However, the approach was still restricted to error correction codes of short length, limiting the applicability to real-world PUF scenarios. The approach was, therefore, further extended to compute the conditional min-entropy of a post-processed PUF response also for realistic code length, as discussed in Section 10.2.2.

6.3.5 Systematization of PUF Metrics

After introducing the metrics in the previous sections, this section provides a systematic to categorize them and draws conclusions about the state of the art.[7] Fig. 6.2 – called metric landscape – gives an overview of the classification. The landscape is partitioned according to the four dimensions mentioned earlier:

Upper-Left: In the upper left part, the metrics are listed that operate on a single chip in order to evaluate the PUF quality. I.e., they can provide information if a defect has been found for a specific chip.

Upper-Right: In this part, metrics are listed that operate on fixed positions but over multiple chips. These are capable of identifying weaknesses at specific positions of the chip.

Lower-Right: Metrics that identify weaknesses regarding the challenge response behavior are listed here. These operate on fixed positions and circuits and compute evaluation results across multiple configurations. Machine-learning-related evaluation strategies are omitted.

Lower-Left: In the lower left, reliability metrics are listed, evaluating the PUF over time.

In addition, the landscape has inner and outer parts. In the dark-shaded outer parts, correlation-related metrics are presented; in the inner parts, bias-related metrics are provided. Metrics listed

[7] The entropy test on the secret is omitted since it does not operate only on PUF data but requires a complete system and knowledge about the applied code.

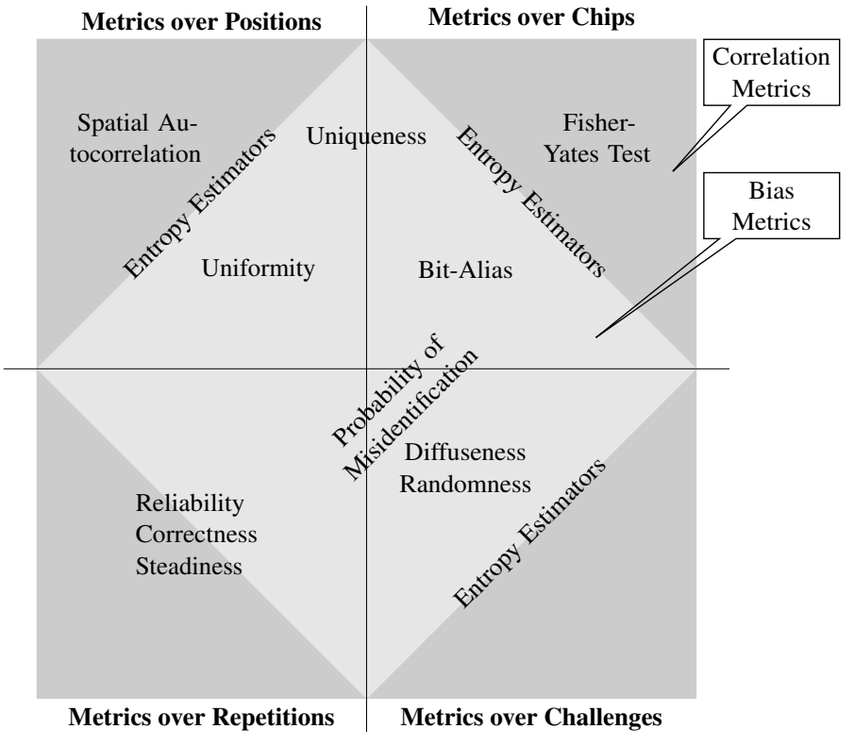


Figure 6.2: Classification of state-of-the-art PUF evaluation metrics.

multiple times or written across borders of the quadrants implicitly or explicitly evaluate the quality with respect to several effects.

For example, joint entropy – or generally an entropy estimate – is usually estimated, e.g., via CTW, by compressing all available data from multiple chips. This way, all effects contribute to this metric. Uniqueness cannot reliably cover correlation effects but is sensitive to cases where bias is observed in the data. In contrast, Bit-Alias is measured per position, so it is not suited to provide information about the quality of individual chips.

Given the landscape, it is evident that there are lots more metrics covering bias effects than correlation effects. Nevertheless, it seems that – besides correlations over challenges – predictability with respect to bias and correlation effects is nowadays well covered. The landscape also reveals that several tests are redundant and can be omitted. Of course, the selected tests should then be enriched by hypotheses or significance tests, which is today not always the case.

Reliability and correctness are written across the border of correlation and bias in the reliability quadrant. The reason is that correlation in the time domain is interpreted for the landscape as a systematic effect like a temperature drift. These tests can provide reasonable reliability estimates when using a proper reference and appropriate data.

Overall, the picture shows that reasonable tests for the PUF quality have been established over the last years. However, it might be noted that the tests discussed so far do not cover higher-order alphabet PUFs (cf. Section 10.1). Also, no analysis of analog PUF responses has been discussed so far, for which a summary is given in Section 6.4.

6.3.6 ISO/IEC 20897 – A Current Standard for PUFs

Recently, the standard *ISO/IEC 20897* has been published, which – in its second part [1] – also makes some suggestions regarding PUF testing. The corresponding appendixes and concrete

examples in the standard are explicitly of informative nature only. Nevertheless, this section aims to relate this standard to the tests mentioned in this collection.

The first property tested by the standard is Steadiness. The concrete suggestions in [1] in this regard is to test for BER as the portion of errors in a response and analyzing the distribution of the intra-chip hamming distance. The standard also suggests a stochastic model for Steadiness based on noise measurements, but it is not apparent how it shall be applied for evaluation.

Regarding the test of unpredictability, the standard first suggests a test for randomness. For this purpose, different statistical tests and entropy tests for TRNGs are mentioned. These tests should be applied – according to the standard – in the direction of the positions on a chip and in the direction of challenges. As mentioned earlier in this collection, concatenating bits without a clear strategy might lead to wrong security claims. One suggestion in the standard's appendix is concatenating for each fixed position on a chip (i) all columns over challenges and (ii) all rows over chips for this position. This results for N_p PUFs on a chip in $2 \cdot N_p$ data sets. As an example, for a test method NIST SP 800-90B is mentioned where the use of the data sets as restart data is implied. The standard also discusses that this test cannot be done if an insufficient number of PUF responses is available.

As a last unpredictability test, a uniqueness test is suggested, which is, e.g., the one indicated by [67]. Alternatively, again, some entropy test for TRNGs is suggested.

Overall, the standard heavily relies on TRNG tests. Dedicated PUF tests play a secondary role. This might be sufficient in several cases to disqualify PUFs which provide non-iid responses. However, there are some indicators that these tests fail to find critical effects in PUFs, like spatial correlations. Also, PUFs with sufficient quality to be used might fail in tests designed for TRNGs since the specific requirements of PUFs are not considered. Nevertheless, the main suggestion of the standard, namely to test the different dimensions of a PUF separately, is in line with the state of the art previously presented.

6.4 Remark on PUF Evaluation in the Analog Domain

Providing tests for digitized data of PUFs as in the previous sections is a reasonable approach for most scenarios: These data might also be available in designs that are ready for use and will likely be one base for PUF certification. However, in order to improve PUF designs, tests on analog PUF responses are useful, too, although they are only available for specific PUFs.[8]

Testing analog PUF responses is mostly more straightforward when compared to testing the finally quantized responses. Examples of test strategies include computations of mean and variance. Also, the application of tests, like the Anderson-Darling-Test to check if data is drawn from the expected distribution, is useful in this regard.

In order to identify spatial correlations, Moran's I and Geary's c – previously mentioned in Section 6.3.3 – have been suggested in **[165]**. Another considerable suggestion is the application of a Principal Component Analysis (PCA). **[166]** has shown that with the use of a PCA, PUF-relevant defects like gradients on a chip can be detected. However, overall, the PUF evaluation in the analog domain serves more the purpose of improving a design than to evaluate a PUFs quality.

[8] Recall, that in this collection the term analog PUF responses is used for digitized analog data like counter values, used for measuring the frequency of a ring-oscillator.

7. Attacks on Hardware-Intrinsic Security

All kinds of attacks that are relevant to classical cryptographic algorithms must be considered for hardware-intrinsic security mechanisms, too. However, due to their specific nature and application, (i) some attacks are of particular relevance, and (ii) some additional attacks must be considered. This section provides an overview of such attacks. In this, the attack vectors are mainly of relevance for PUF technologies, in particular for actual PUFs and PUF-related tamper protection techniques. Only in section Section 7.2 some insights related to TRNGs are provided.

7.1 Helper Data Related Attacks

Helper data are usually used if PUF technologies are applied for key storage. Looking at Fig. 5.1 on Page 33, it becomes apparent that every element in the chain from the raw analog PUF response $\mathbf{p}_{raw}$, over the quantized PUF response $\mathbf{p}$ and the derived codeword $\mathbf{c}$, to the secret $\mathbf{s}$ and finally the key $\mathbf{k}$ must be protected. Helper data are used in this process in the context of quantization (cf. Section 4.4) and within a HDA to map a PUF response to a codeword (cf. Section 5.1.1). Therefore, these steps can potentially be targeted when applying attacks related to helper data. Different levels of attacks related to helper data must be distinguished:

Passive Attacks: This kind of attack observes the helper data. By analyzing the statistics of the helper data or by using additional side knowledge on the statistics of the PUF, an attacker can get an advantage in guessing the PUF response or a secret derived from it.

Direct Aktive Attacks: Depending on the HDA used, an attacker can gain some information regarding the secret just by manipulating helper data and observing if the output changes. This can be interpreted as a FIA.

Aktive Attacks as Enabler: In particular for several SCA attacks in Section 7.2, the attacker requires some possibility to observe data dependency. Manipulation of helper data is used for this purpose.

Passive attacks based on helper data are usually not directly elaborated. Instead, in the literature, the leakage properties of helper data, which lead to entropy loss of the PUF response as well as of the finally derived secret, are discussed (e.g., [34]). Helper data leakage can – depending on the HDA, cf. Section 5.1.1 and **[122]** – completely be avoided if a PUF provides iid unbiased response

bits. Since this is usually not the case, in reality, helper data leakage is a severe issue and can hardly be avoided. However, since helper data leakage usually provides only probabilistic information, reducing the entropy and thus the guessing effort for an attacker, it can be countered up to a certain extent relatively easily: Increasing the secret length – i.e., the number of PUF bits and all related quantities in the postprocessing in Fig. 5.1 – also increases the entropy for an attacker. The key with sufficient entropy is derived from this long sequence by compression.

Direct attacks exploiting helper data are described, e.g., in [61, 37, 38, 34]. *Helper data manipulation attacks* are usually based on the following idea: The attacker has the hypothesis that a modification of helper data causes some error. This error is observable by detecting if the reconstructed key is correct; an observation which can be expected to be always feasible. The gained information is used to verify the hypothesis so that the attacker learns from the observation.

To prevent such attacks, two strategies are reasonable:

1. *Preventing helper data manipulation* by using write-protected on-chip NVM to store helper data would obviously hinder the described active attacks. However, in a PUF scenario, one might reasonably assume that there is no secure NVM. Available storage technologies – in particular in a lightweight scenario – are, therefore, expected to be limited to rather low-security solutions like fuses. While using such technologies would make the attack much harder,[1] storing hundreds of helper data bits with such a technology seems impractical.
2. *Preventing the attacker from gaining knowledge* seems the more practical approach. The most straightforward strategy in this direction is to hash the secret after error correction with the helper data using a cryptographic hash function. The idea was first suggested in [61] and causes any manipulation of the helper data resulting – with high probability – in a wrong key. Thus, an attacker cannot learn from the observation if the secret derived from the PUF under a helper data manipulation remains the same or not. This efficiently avoids the exploitation of the attacker's hypothesis. Please note that for this approach to be effective, the attacker must not be able to find a second set of helper data that causes – together with the unknown secret – the same key. It is worth mentioning that the hashing of the helper data with the secret can be merged with a compression required due to helper data leakage or an imperfect PUF so that it results in low hardware overhead in practice.

While with the second strategy, a practical method is available to counter passive and direct active attacks, the active attack as an enabler for other attacks remains. This attack is critical since it might be used in combination with SCA as described in Section 7.2. It will become apparent below that the attacker does not rely on the fact that the key is correctly derived in that case. Thus, such manipulations cannot be hindered by ensuring that the key definitely changes if an attacker tampers with the helper data. Also, authentication of the helper data is not achievable if the PUF stores the key needed for authentication or to decrypt the authentication key. Finally, it is hardly possible to limit the number of manipulations and destroy a device if it is under attack: First, counting the number of manipulations implies non-volatile storage of a counter value that cannot be reset by an attacker. If there is no secure NVM in a PUF scenario, only solutions like fuses might be reasonable, limiting a practical count of failed reconstructions without destroying the devices to a low number. Since reconstructions can also fail due to errors in the PUF and, e.g., too extreme operation conditions, it is a design question if this is a feasible solution. Second, an attacker does not necessarily need to wait until the key is reconstructed. It is, therefore, not sufficient to increment a counter only if a key reconstruction fails. However, triggering a counter incrementation in any case a key is reconstructed, and decrementing the counter – or removing the trigger – only if the key reconstruction is successful, would require some non-volatile on-chip memory, which can be frequently set and reset but cannot be manipulated by an attacker. Such a memory technology might not be available in the PUF scenario. Third, some attacks manipulate the helper data in a way so

[1] An attacker would require, e.g., to open the chip and to use a FIB to reset fuses.

that the reconstructed key does not change, so that only a combination of hashing the helper data and a protected counter can hinder them.

It can be concluded that in case the helper data manipulation is used as an enabler for another attack, only two options are practical: The manipulation of helper data can be hindered completely, which is hard to achieve in certain scenarios, as discussed before, or the attack enabled by the helper data manipulation must be prevented, e.g., by removing side-channel leakage.

7.2 Side-Channel and Fault Injection Analysis

When hardware-intrinsic features are used for security mechanisms of devices that are accessible to an attacker, hardware-related attacks must be considered. Of particular interest in this context are attacks based on SCA and FIA. Two types of attacks are distinguished in the following since they have fundamental differences: (i) Attacks on the measurement apparatus, e.g., a PUF or TRNG primitive and (ii) attacks on the postprocessing.

7.2.1 Side-Channel Analysis of the Measurement Apparatus

SCA attacks have been presented on TRNGs as well as on PUFs. In particular, the following primitives have been attacked:

Ring Oscillator

ROs have been analyzed by SCA first in the context of PUFs in [107] using an electromagnetic (EM) side channel. The general idea behind this kind of attack on ROs is to measure power or EM traces to receive the power spectrum in the frequency domain.[2] Peaks in this spectrum indicate the oscillation frequencies of the ROs. [107] used this to attack an RO PUF with sequential overlapping readout of frequency pairs, i.e., a comparison of the pairs $(RO_A, RO_B), (RO_B, RO_C)$, etc. with frequencies $(f_A, f_B), (f_B, f_C)$. Since at every point in time, only two oscillators are active, the corresponding two frequencies can be identified in the spectrum. In addition, since one RO (RO_B in the example) is reused, and the order of the readouts must be constant and, thus, be assumed to be known by an attacker, an attacker can reveal the secret from observing the active frequencies over time. The conclusion from this analysis is that an overlapping comparison of ROs in an RO PUF leads to a vulnerability through SCA.[3]

A detailed analysis of RO PUFs using localized EM measurements in [106] revealed that the observed leakage from the RO PUF is mainly related to the counter and partially also to the multiplexing to these counters. As a consequence, an attacker can predict secret bits derived from an RO PUF if pairs of ROs are compared with distinct counters, and these counters can be spatially resolved. This resulted in different options for more securely implementing an RO PUF: It has been suggested to interleave the counters or to randomly permute the counters. This way, it is not possible for an attacker to predict which frequency belongs to which RO and thus to derive the secret bit, which is usually the sign bit of the difference (cf. Section 4.2.2). [138] supports the requirement of hiding the order of frequencies by showing that close-by placed counters can still be spatially resolved using localized EM analysis.

Remark: It is worth noting that the mentioned hiding mechanisms protect only the sign of a frequency difference, not the frequency difference itself. Thus, these methods are not sufficient if a quantization of frequency differences into HOA symbols is to be achieved, cf. Section 9.1.

Besides the SCA using the electromagnetic emanation or the energy consumption as a side channel, Laser Voltage Probing (LVP) has been used as a side channel for attacking RO PUFs

[2] Most commonly, measurement is done in the time domain, and a Fast Fourier Transform (FFT) is used to transform into the frequency domain.

[3] Please note that even without the existence of such a side channel, an overlapping pairwise comparison is suboptimal since it causes correlations of the PUF responses.

in [92]. This is particularly useful if multiple ROs run at the same time, and they cannot be resolved with power or EM SCA. The suggested process for this kind of attack is a three steps approach: First, the attacker approximates the range of the frequencies of interest with power or EM SCA. Second, they apply Laser Voltage Imaging in order to identify the nodes where leakage can be observed. Third, these ROs are probed individually using LVP. Since LVP allows for probing of single nodes of the circuit, the attacker can resolve individual frequencies and read the secret in this way. The only countermeasure against this kind of attack suggested in [92] is to sense unexpected shifts of single RO frequencies since these can indicate an LVP attack on the system.

While SCA attacks on ROs have been mainly suggested in the context of PUFs, it is worth mentioning that [55] considered such an attack also in the context of a COherent Sampling ring Oscillator (COSO)-based TRNG [84]. A COSO-based TRNG consists of two ROs, one sampling the other. A Toggle Flip Flops (TFFs) – representing the Least Significant Bit (LSB) of a counter counting the number of transitions of one RO – is used in this construction to generate the secret. While one might expect that – like for PUFs – the spectrum is attacked in this case, it turns out that such attacks were not successful. Instead, the reset of the TFF generating the secret was targeted with localized EM, revealing which bit was stored.

Transient Effect Ring Oscillator:

The Transient Effect Ring Oscillator (TERO) was introduced as a TRNG in [158] and later suggested as a PUF primitive in [14]. While the designs slightly differ, it is best explained on a high level by today's usually considered version in [14]: A bistable ring[4] is brought into an unstable state. After releasing an enable signal, the ring starts oscillating and ends after some time in a stable state.[5] The number of oscillations it takes until the ring settles is measured by counting the number of periods at some output node of the bistable ring. The expected duration until the ring settles is defined by the design of the ring and by process variations around which the actual counter value fluctuates due to noise. The LSB of the counter, thus, depends on noise and can be used as a random number, while the Most Significant Bits (MSBs) are device-specific and can be used to form a PUF.

The first attack mounted on this primitive was demonstrated in **[152]** with a parallel work providing similar insights in [113]. These attacks targeted the estimation of the counter value's MSBs to show that protection mechanisms are required when using a TERO as a PUF. While it was considered infeasible before to predict the counter values of a TERO PUF due to short oscillation time and small differences between oscillations, both works have shown with slightly different methods that the relation between counter values generated by a TERO can be resolved with similar techniques as they are used in the RO context and some additional processing of data. Thus, the same strategies regarding the comparison of counters discussed in the context of ROs must be considered. A successful attack using localized EM resolving close-by counters has not been shown in this context, but it is plausible that the same attack strategies discussed in the context of ROs are feasible, too.

While the attacks on the PUF primitive did not need to resolve the LSB, this was targeted in an SCA of the TERO used as TRNG in [116]. In that case, a TFF replaced the counter so that only the noisy LSB is generated. This TFF was targeted with localized EM measurement to reveal the random number. The authors observed exploitable leakage due to the reset of the TFF but did not show a vulnerability related to the oscillation of the TERO.

Arbiter PUF:

An SCA attack on the Arbiter PUF was suggested in [108], and related attacks were presented, e.g., on a simulative level in [86] and using actually measured data in [3]. [86] suggested two attack points: (i) an attack on the latch at the output of the Arbiter PUF, i.e., the arbiter, and (ii)

[4] This means a ring with an even number of inverting elements

[5] In practice, the ring can oscillate for a long time and is interrupted after a fixed time if not settled.

an attack on a subsequent Flip Flop (FF) storing the arbiter's decision for further processing. By simulation, the authors of [86] have shown that leakage from both points can be exploited where the attack on the FF shows higher SNR. The actual attack in that work was done by machine learning: The response-dependent leakage profile was learned on a reference PUF, which is under complete control for an attacker; The model was successfully used to predict the response of other simulated PUFs afterward.

The analysis in [3] targeted an Arbiter PUF used within an Interpose PUF [115]. The authors measured the PUF under 100,000 challenges and repeated every measurement 1,000 times to increase the SNR. With that measurement set, they received a power distribution, from which challenges can be identified as corresponding most likely to logical ones and, respectively, zeros for the response. With this information, the authors trained a Linear Regression (LR) model, which allowed them to satisfactorily predict the arbiter and, eventually, the Interpose PUF.

Another approach suggested to attack the Arbiter PUF by [146, 145] was an optical readout. The authors have shown that photon emission due to switching activities can be exploited. The authors had to read the PUF under the same challenge multiple times in order to collect enough photons. They demonstrated that the information they received this way suffices to reveal the PUF response.

It has to be noted that for the mentioned attacks to be practical, repeated measurements of the PUF under the same challenge were needed. This is not possible with a challenge-sharing strategy as outlined in Section 5.2.1. Nevertheless, the results demonstrate another time the vulnerability of PUFs.

SRAM PUF

SCA was also applied on SRAM PUFs in [56]. In this paper, the light emission caused by the switching activity of an SRAM cell's transistors was observed, similar to the previously mentioned attack on Arbiter PUFs that was presented later. As a result, the authors were able to read out the secret from an SRAM PUF. The attack required thinning of the chip and was demonstrated on a 600 nm technology. Within this technology, the authors were also able to edit the SRAM cells using a FIB and to enforce the switching of SRAM cells into a specific state at power on.

7.2.2 Side-Channel Analysis of the Postprocessing

The attacks in Section 7.2.1 target different PUF or TRNG primitives and are therefore specific for the corresponding measurement circuits. However, the secret bits derived from a PUF or a TRNG are frequently post-processed before they are used in some cryptographic applications. Regarding the postprocessing of TRNGs, it is worth noting that successful attacks need to reduce entropy from a single measurement trace. Nevertheless, examples like [114] indicate that such attacks might be feasible under certain circumstances. In that publication, the SCA was mounted on the output register of the Random Number Generator (RNG) of an STM32 microcontroller. This is similar to the attacks mentioned for TRNGs in Section 7.2.1 but shows that also Pseudorandom Number Generators (PRNGs) must be implemented with SCA in mind as discussed, e.g., in [18].

Regarding PUFs, postprocessing is mainly relevant in the context of key storage. In that context, two attack points are of relevance (cf. Fig. 5.1): SCA can be mounted on the error correction decoder or the security amplification step. It is usually not of relevance to target the key enrollment step since this shall be done only once and ideally in a trusted environment. The HDA is usually implicitly considered, as discussed below.

Starting with the security amplification step, it has to be noted first that, at this point, a noise-free secret is compressed. For this purpose, usually a hash algorithm is used. Therefore, the same attacks that are possible on the hash algorithm in any other security context must be considered in the PUF context, too. This has been demonstrated in the example of a Toeplitz hash in [108], which was suggested for a PUF design in [95]. The Toeplitz hash in that scenario takes single secret

bits, which are derived from a PUF, and XORs – depending on the bit value – a specific row of the Toeplitz matrix or an all-zero vector onto the state of an accumulator. As a consequence, the energy consumption of the operation depends on the secret bit and is observable in a power or EM measurement trace. A successful attack was, thus, feasible with a Simple Power Analysis (SPA) after averaging over a few hundred measurement traces.

While attacks on the security amplification step are strongly related to attacks explored in the cryptographic context, [28] also suggested the idea to mount attacks on the error correction decoder. A first practical SCA on BCH and Reed-Solomon (RS) decoders was demonstrated in [79]. It targeted software implementations and suggested two attacks: (i) The initial software under attack used a conditional execution depending on symbols or bits in the decoder input being zero. For the BCH code operating on bits, this information directly leaks secret information; for the RS code, helper data manipulation was used to enforce individual symbols becoming zero, which again provides information about the secret symbol. To retrieve the corresponding information, SPA suffices. (ii) The work has further shown that even if the data dependence in the first attack is circumvented, a template attack is possible. For this purpose, templates for decoder inputs were built, and inputs with well-distinguishable templates were selected. To attack a specific device, helper data of the device under attack were manipulated (permuted) to enforce the inputs that correspond to the inputs of the templates, and a comparison of the templates and the observed energy consumption allowed to derive the secret input symbols of the decoders.

Another approach was taken in [109] to mount an SCA attack on an error correction decoder in hardware. In that work, a code concatenation of repetition and BCH code was analyzed. The attack utilizes that the PUF responses are bitwise fed into an input register of the outer BCH code. When observable, the energy consumption of this register leaks the HD between two subsequent secret codeword bits. For the purpose of this attack, the helper data is considered as data, and the PUF response is the secret information under attack. Under this setting a Differential Power Analysis (DPA)-based [83] Correlation Power Analysis (CPA) [15] was used to reveal the secret information: Helper data are manipulated to enable the attack as described in Section 7.1. Knowing which n helper data bits belong to a repetition codeword[6] and taking the PUF response to be constant for all decodings, it is known that the output of the repetition decoder remains constant as long as less than $n/2$ helper data bits are flipped and changes its bit value otherwise. As a consequence, a difference in energy consumption under different manipulated helper data inputs should be observable, and – since the repetition decoder processes small codewords – a divide-and-conquer strategy is possible. The hypothetical energy consumption for the CPA is assumed to be lower for two equal subsequent output bits of the repetition code arriving at the BCH input than for two unequal bits. By manipulating the helper data of single repetition codewords and, thus, influencing the output of the repetition decoder and by correlating hypothetical energy consumptions with observed power traces, [109] has shown that SCA indeed reveals the secret. The attack has been improved in **[153]** and was shown to be feasible even without using the repetition code in [148].

Notably, the attack is not detectable and feasible as long as helper data manipulation is possible: Although the attacker manipulates helper data of the repetition code and introduces an error at the BCH input, the decoder is most likely designed in a way so that a single additional error at the BCH input under nominal operation conditions does not cause a wrong key that could be detected. The existence of an error introduced after the repetition decoder through helper data manipulation cannot be exploited to detect an attack either since it cannot be distinguished from regular errors. Thus, another strategy to counter the SCA attack is needed. [109] suggested codeword masking: A random codeword – generated by encoding a random number first with the outer BCH and then with the inner repetition code – is computed and XORed onto the codeword derived from the PUF. Since the codeword is unique for every decoding, the described CPA is no longer possible, but

[6] n is assumed to be odd.

higher-order DPA would be needed to succeed. The described strategy of codeword masking is applicable not only to the specific case of a code concatenation of repetition and BCH code but to any linear code.

7.2.3 Fault Injection Analysis

Besides SCA, FIA has been discussed in the context of hardware-intrinsic security. In terms of TRNGs, the goal of such attacks is usually to eliminate noise effects in order to make the RNG output predictable. In particular, a list of locking attacks on oscillator-based TRNGs has been proposed [101, 11, 112]. These publications provide analyses of the problem that oscillators tend to lock to another nearby frequency.[7] While most publications focus on classical ROs, [112] has shown that also TRNGs based on TEROs or selfe-timed oscillators [26] are affected.

In parallel to attacks on TRNGs, attacks on PUFs have been presented. [147] suggests attacks on XOR Arbiter PUFs and RO PUFs. Laser fault injection is used in both cases to manipulate the output of the overall PUF. In the case of the XOR Arbiter PUF, this allows for observing the output of a single Arbiter PUF instead of the XOR of several PUFs, making machine learning much easier. For the RO PUF, the suggested fault injection stops the oscillation of the ROs so that the counter values counting the periods within a fixed time have the same value, and the output bit is set to an internally defined default value. Depending on the concrete implementation, knowing this value or even knowing that the difference of the counter values is zero enables attacks to guess the secret output bit of the PUF. Bringing, e.g., the error correction to the limit of its correction capability so that any further error would cause a wrong key and enforcing an additional bit to a default value reveals a secret bit: Suppose the default value caused by manipulation is equal to the bit value without manipulation. In that case, the derived key will be correct (and incorrect otherwise), which is information observable to an attacker.

Another manipulation-related attack using an injection-caused side channel is presented in [93]. This work does natively not target a PUF but battery-backed SRAM; However, it claims that the attack is feasible on SRAM PUFs. To reveal a secret from SRAM, the authors use a laser beam causing during a scan thermal laser stimulation. This causes, depending on the state of the SRAM cell and the currently scanned transistor, additional current flow. Observing this current flow reveals the state of the SRAM cell and, thus, the secret stored by the device.

Overall, the examples shown in this section demonstrate that fault attacks are possible for PUFs. However, they are by far less explored than SCA. In particular, neglecting helper data manipulation attacks as a form of FIA, fault injection attacks have been mainly mounted on TRNGs and PUF primitives but not on the corresponding postprocessing. An advancement in this direction is presented in Section 9.2.

7.3 Machine Learning Attacks on Physical Unclonable Functions

One of the most prominent attack vectors to PUFs is Machine Learning (ML). This kind of attack is relevant for Multi Challenge PUFs since usually the challenge-response behavior shall be predicted.

Challenge-Response Driven Machine Learning Attacks

The classical attacks in this domain operate on the challenge-response behavior of PUFs. This involves machine-learning strategies like Support Vector Machines, Logistic Regression, or Evolution Strategies [132]. These attacks are, in particular, well suited if a model of the PUF can be provided and parameters must be fitted for the model. If PUF constructions get more complex, a tendency towards the use of Deep Learning strategies is observed [135, 170] and such strategies have been applied successfully to model, e.g., Multiplexer PUFs [134] or Interpose PUFs [115]. For machine

[7] The frequency the oscillator locks to can also be the harmonic of an injected frequency.

learning methods, it has been shown that the learning effort for Arbiter PUFs is polynomial in the challenge length [49]. Also, the learning effort for XOR Arbiter PUFs, while exponential in the number of XORs, can be considered to have a polynomial effort in the challenge length since noise effects limit the practical recombination of responses through XORs to a low number [48].[8] [47] provides a tool to evaluate the quality of PUFs w.r.t. machine learnability.

Machine Learning Attacks using Soft Information

Another branch of ML attacks on PUFs considers soft information. This information can be gathered in the form of reliability information from repeated measurements [12] or by combining machine learning with SCA [36, 86]. The benefit of reliability information becomes apparent if a linear PUF model is considered: The reliability information can be interpreted as a distance of how far the analog value quantized by the PUF to a response is away from some decision boundary. The output symbol states on which side of a decision bound – e.g., towards a logical one or zero – the analog value is located. Considering the challenge bits as weights and the measured process variations – e.g., the delays of a linear model of an Arbiter PUF – as free parameters, a linear equation system can be formed and solved, resulting in the delay parameters, or more precisely a scaled version of the delay parameters.

In practice, the accuracy of the measured reliability information is limited so that the solution can only be approximated, for which more equations are useful, and an ML approach can be used to approximate the solution. Furthermore, reliable information has been shown to be also valuable for attacking more complex, e.g., XOR Arbiter PUFs [12]. It is, however, worth recalling that reliability-driven attacks, which retrieve the reliability information from multiple readouts of the same challenge-response pairs, are hindered by challenge-sharing strategies as described in Section 5.2.1. In addition, the results in [12] suggest that the effort in terms of required challenge-response authentication runs to sufficiently approximate the reliability information can be – in particular for reliable PUFs – so large that a classical challenge-response driven attack might be more efficient.

A Machine Learning Attack on a PUF-Based Key Store

Besides attacks on challenge-response protocols, [12] also suggests a ML based attack on a key storage scenario. For this purpose, they use that for a linear model of a PUF a small change in the challenge would cause only a small change in the response.[9] As a consequence, it is possible to take from a linear PUF model all challenges used to generate responses and to try to partition them such that the HD in each set is minimized. One set is then assigned a logical zero, the other one a logical one, representing the secret PUF response. If the described partitioning perfectly works, the only uncertainty is which partition corresponds to a logical one and which to a logical zero.

The results in [12] have shown that the attack does not work as long as all challenge-response pairs are used. The reason is that there is no obvious partition. However, if unreliable bits are excluded, as is the case in the bit-selection HDA in Section 5.1.1, two clusters are formed, and it has been shown for the case of the index-based syndrome coding that the attack succeeds. In this, the attack is getting more effective as more key bits are generated with the PUF.

While this idea is limited to the specific case of linear PUF models and the use of bit-selection HDA, it still shows that ML attacks on PUF-based key generation must be considered.

[8] While very design specific, it can be expected that at most 10 to 20 responses can be XORed in practice.

[9] A good example is, e.g., a SUM PUF, where a bit flip changes the sign of one out of many frequency differences summed up, and the sign bit of the sum is the secret. For an Arbiter PUF, a single bit flip would lead to a significant change. However, flipping two subsequent challenge bits causes only two delay stages to be different, thus, a small change in the delay of the two competing paths and, consequently, a high chance that the response bit remains the same.

III Contribution

8. Keys from Hardware-Intrinsic Features

It has been discussed in Chapter 5 that PUFs are used to prove authenticity and to store keys with hardware-intrinsic features. In this chapter, first, a new PUF-based protocol to unlock keys remotely is introduced. It uses a Multi Challenge PUF to derive keys, which are intrinsically bound to hardware and are used to unlock functionality. The approach ensures that helper data manipulation attacks (cf. Section 7.1) are not possible, although helper data manipulation itself cannot be entirely prevented. As discussed in Section 5.1.1, helper data start leaking as soon as PUF responses have any statistical defect such as bias or correlations. Since we use a Multi Challenge PUF but do not provide the PUF responses or the key to an untrusted party, this raises the questions (i) if the Multi Challenge PUF can be machine-learned without knowing responses and (ii) how many challenge-response pairs can be used if ML is possible. This question has been targeted in the second publication introduced in this chapter, with the result that the ML based on helper data is indeed possible and that the difficulty to learn a PUF in this context does not only depend on the strength of the PUF but also on the used code. Since polar codes make machine learning harder when compared to other codes usually used in the PUF domain, and since polar codes have been suggested before but without actual hardware implementation, the last section of this chapter shows that polar codes can be implemented for a PUF scenario in hardware with reasonable effort.

8.1 A Method to Derive Multiple User-Dependent Keys from a PUF

The contribution introduced in this section targeted a low-cost system, e.g., an IoT gateway with trusted hardware. An on-board Multi Challenge PUF is assumed to be characterized within a trusted environment. On the gateway, potentially untrusted software is running. The gateway has access to several actuators and sensors. Only a legitimate user shall be able to access these actuators and sensors, and confidential and authenticated data transmission shall be possible. In this, the gateway is assumed to be not able to perform any asymmetric cryptography. The approach in **[45]** solves this problem with the help of a PUF and Authenticated Encryption with Associated Data (AEAD).

8.1.1 Proposed Method and Results

The described concept is based on multiple user-specific pre-shared keys, which are stored with a Multi Challenge PUF on the gateway and enrolled to legitimate users through a trust center. A user trying to connect to some actuator or sensor through the gateway sends its ID and the helper data to reconstruct the user-specific key from the PUF. In the same message, also – possibly confidential – commands to be executed or data can be sent. An AEAD scheme is used to authenticate all of these data and to encrypt, e.g., the command with the user key. The gateway can reconstruct the user key by using the PUF and the authenticated helper data. With this key, the authenticity of the transmitted data is tested. The command is only decrypted and executed or forwarded to an actuator or sensor if the authentication passes.

■ **Result 8.1.1** The designed concept has shown that the combination of an AEAD scheme with a PUF is indeed useful: The helper data must be provided in an unencrypted form to be used to derive a key. However, since it is associated data, manipulation of the helper data is detected by the device.

The work also includes a comparison of different hardware implementations of PUFs and AEAD schemes. Considering the overhead required for a PUF including error correction, the PUF dominates the latency, which is in particular due to the necessary error correction decoding. With respect to area, the AEAD schemes tend to show a similar or larger footprint compared to the PUF, including an error correction decoder in hardware. ■

8.1.2 Bibliographic Information

This work **[45]** appeared as:

Christoph Frisch, Michael Tempelmeier, and Michael Pehl
PAG-IoT: A PUF and AEAD Enabled Trusted Hardware Gateway for IoT Devices
In 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI),
Pages 500 – 505. IEEE, 2020,
doi: `https://doi.org/10.1109/ISVLSI49217.2020.000-7`

8.2 Machine-Learning of PUFs Through Helper Data

In the previously mentioned work, as well as in other publications such as in [156] or **[120]**, a Multi Challenge PUF has been proposed to derive multiple keys from the same PUF. At the same time, it is known that machine learning of Multi Challenge PUFs is possible when knowing their challenge-response behavior. The discussion in Section 7.3 also suggests that reliability information of the PUF might be critical, and [13] has shown that keys can be revealed when certain Multi Challenge PUFs are combined with a pointer-based approach. This raises the question if machine learning is also feasible in the application above and, if so, how many keys can be used before the system is broken.

8.2.1 Proposed Method and Results

In the previous section, an approach was suggested which was based on a Fuzzy Commitment scheme as described in Section 5.1.1. This means the attacker does not have access to the secret PUF response, and helper data do not reveal any reliability information. In addition, potentially unreliable responses are used, too, hindering the clustering of challenges and, thus, also the attack in [13]. The proofs in several previous information theoretic works have also shown that helper data do not leak about the secret for iid, bias-free PUF responses. But a Multi Challenge PUF provides non-iid, i.e., correlated, responses, resulting in some secrecy leakage. As long as this leakage is limited, an assumption might be that compression of data is sufficient to derive cryptographic keys with sufficient entropy. However, there is another critical problem: Existing code-based helper data algorithms, as defined in Section 5.1.1, suffer from privacy leakage [70]. A practical interpretation is easy to get with the help of a 3-repetition code, i.e., a code where one information bit is encoded into three equal bits:

■ **Example 8.1** Assume a PUF providing response bits $\mathbf{p} = [p_0, p_1, p_2]$ and a secret key consisting of one bit $\mathbf{k} = k$. For a Fuzzy Commitment scheme and using a 3-repetition code, k is encoded during enrollment to $\mathbf{c} = [k,k,k]$ and XORed with $\mathbf{p}$ to receive the helper data $\mathbf{w}$. The attacker can observe $\mathbf{w} = [k \oplus p_0, k \oplus p_1, k \oplus p_2]$. The code must be assumed to be known to an attacker. Thus, the attacker knows that all codeword bits are equal and XORing helper data bits results in an XOR of responses, i.e., $w_i \oplus w_j = p_i \oplus p_j$. In other words, the attacker does not learn the PUF responses, but the relation of the PUF responses. ■

The case in the example can be generalized for all linear codes and code-based helper data algorithms as defined in Section 5.1.1: For constructions such as Syndrome Construction and Systematic Low Leakage Coding, the helper data is generated such that it is an XOR of PUF responses, and XORing helper data can be used to receive additional dependencies. For constructions such as Fuzzy Commitment or Code-Offset Construction, where PUF-independent randomness is used, targeted recombination of helper data bits cancels this randomness. An attacker remains in any case with *XOR equations* of helper data that corresponds to the XOR of PUF responses, as shown in the example.

If the PUF is a Multi Challenge PUF, the resulting XOR of responses received from XORing helper data can be seen as the response of a new, more complex PUF and is a function of challenges. The XOR equation of the responses of a PUF under two different challenges becomes in the example of a three repetition code

$$w_i \oplus w_j = p_i \oplus p_j = \mathrm{puf}(\mathbf{ch}_i) \oplus \mathrm{puf}(\mathbf{ch}_j).$$

It is known that for a Multi Challenge PUF, $p_i = \mathrm{puf}(\mathbf{ch}_i)$ can be machine-learned given a sufficient number of challenge-response pairs. This is exploited in our research: A specific machine-learning model, a Siamese Neural Network (SNN) [102], is used for this purpose. It can be considered multiple neural networks, each modeling the same PUF, plus an additional layer recombining

the responses. This way, the input consists of two or more different challenges, and the output represents the difference in – i.e., the XOR of – the responses for these challenges. Given such a model, it is evident that it can be trained with the XOR equations defined above when the challenges are considered the features and the response differences the labels.

Although it is evident that a model can be trained, it is not yet apparent how an attacker can exploit this since all PUF response differences are known anyway and can be computed directly. In the example above, e.g., one secret bit was stored with the PUF, and the attacker who knows all differences has one bit of entropy left since they must guess one bit to conclude on the value of all PUF responses, which is equivalent to guessing the secret bit directly. However, the SNN model is more powerful: Given, e.g., z 3-repetition codewords storing z secret bits with $3 \cdot z$ helper data and all PUF bits derived from the same Multi Challenge PUF, the attacker can train the SNN using the dependencies revealed by each codeword independently. The attacker has no information at this point regarding the dependence of responses between different codewords. However, after training the SNN, the attacker can query the model of how the responses of codewords between challenges depend on each other. If the model is sufficiently trained, the model will reveal this information, leaving the attacker with an uncertainty of one bit in this case since they need to guess one response as a reference. The general concept described cannot only be applied to guess unknown dependencies between PUF responses used to store different codewords but also to predict unknown dependencies between PUF responses used in the same codeword.

It should be noted that in the scenario of a PUF key storage, the attacker does not need to predict the PUF response in this scenario perfectly. Since the key is reproduced by use of an error correction code, the attacker can accept up to a certain error when guessing the PUF and can use the known helper data and error correction code to get rid of errors in their guess and to reveal the correct secret key. It is also worth noting that the data an attacker learns are not noisy since they are computed during enrollment and fixed afterward.

■ **Result 8.2.1** As a result, our work shows that the described approach is indeed able to break a PUF key storage solution only by knowing challenges and helper data. In summary, the key findings from applying this attack strategy are as follows:

1. Simple Arbiter or SUM PUFs cannot be used to store a secret key since they can likely be learned already from the helper data of the first key so that the key can be immediately revealed.
2. A more complicated PUF makes machine learning harder, and the results suggest a similar increase in difficulty as is the case for classical challenge-response-based machine learning.
3. A more complex code, i.e., a code forcing the attacker to form XOR equations with many XORs, makes machine learning harder, which corresponds to intuition. However, it also says that the most frequently used inner code for PUFs, namely the repetition code, results in a higher vulnerability.
4. A code that allows establishing multiple XOR equations with only a few XORs each is also beneficial for an attacker since more learning data are available, which means that, e.g., longer repetition codes are more vulnerable than short ones.
5. It follows from the previous points that a sufficiently long-term secure key storage approach can only be implemented with a Multi Challenge PUF if the number of challenge-response pairs used is limited to a fixed, sufficiently small value.

■

8.2.2 Bibliographic Information

This work **[140]** appeared as:

Emanuele Strieder, Christoph Frisch, and Michael 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,
2021(2), Pages 1 – 36
doi: https://doi.org/10.46586/tches.v2021.i2.1-36

8.3 On the Feasibility of Implementing Polar Decoders for PUFs

The previous section has shown that it might be beneficial to use a complex standalone code in order to prevent specific attacks. This finding is confirmed by the entropy analysis for the key after error correction in Section 10.2.2. One candidate solution for this purpose is a Polar Code. Polar codes have been suggested for the domain of PUFs in [21, 54]. However, there is a reason why practical implementations are mostly [59] based on repetition and BCH codes: As outlined in Section 5.1.2, concatenated codes have been introduced to PUFs to reduce complexity. In particular, if dedicated hardware accelerators are used, repetition and BCH codes can be implemented with comparably low latency and low area footprint. However, it was not researched before if Polar Codes can be implemented in a way such that they are suited for the lightweight implementations that are typically targeted by the PUF technology. The following contribution closes this gap.

8.3.1 Proposed Method and Results

Polar Codes are most efficient if soft information is used for the decoder. Soft information is not available for all PUF types. The reason is that, on the one hand, soft information stored in helper data might leak additional information, and possible security threats introduced by such helper data must be carefully considered. On the other hand, generating soft information on the fly is not easily possible for all PUF types. In this work, we target an oscillator-based PUF – the Loop PUF – for which a counter difference between two different configurations results not only in a bit value but also indicates the reliability of the response. For this setting, a model similar to [41] is introduced based on which Log-Likelihood Ratios (LLRs) are derived as soft information. Since a limited number of LLRs suffices, they can be computed on the fly from counter differences using a table lookup, fostering an efficient implementation.

With this setting in mind, different designs regarding quantization of the LLRs for different codeword sizes have been studied. For all of them, the goal was to correct an error of 15% per bit in the PUF response down to a block error rate of 10^{-6} for the key.

■ **Result 8.3.1** As expected, a more precise quantization of LLRs – i.e., using more bits for representing the reliability – increases the decoder size but also improves the error correction capabilities. In addition, a larger codeword size improves the error correction capabilities but also increases the decoder size and latency. ■

Based on this analysis, for the implementation, a Polar Code with a codeword length of 512 bits encoding 64 information bits was chosen. I.e., to generate a 128-bit key, the decoder has to be run twice. The LLRs were represented by 4-bit numbers, and parallel processing of eight LLRs was implemented. For the HDA, a Fuzzy Commitment Scheme was used.

The Polar Decoder was implemented for these parameters as a specialized controller processing the different functions required for decoding in the correct order. The resulting decoder was implemented for a 22 nm technology and a frequency of 500 MHz.

■ **Result 8.3.2** The implementation has shown a significant overhead in terms of area and latency of the Polar decoder when compared to a code concatenation of 7-repetition as inner and an $(n,k,d) = (127,64,21)$ BCH decoder as outer code: The area footprint for the Polar decoder is a factor of approximately 1.6 larger, and the latency in clock cycles is increased by a factor of roughly 6.9. At the same time, the number of required PUF and helper data bits to reconstruct a 128-bit key[1] is reduced from 1778 bits for the repetition and BCH decoder to 1024 bits for the Polar decoder.

This result shows that from a hardware perspective, the classical implementation of repetition and BCH code is more efficient. However, the Polar decoder can be implemented at a reasonable

[1] I.e., both decoders were run twice decoding 64 information bits per run.

size, and it is worth considering it depending on the application scenario, given possible security benefits. ∎

8.3.2 Bibliographic Information

This work **[80]** appeared as:

Claus Kestel, Christoph Frisch, Michael Pehl, and Norbert Wehn
Towards More Secure PUF Applications: A Low-Area Polar Decoder Implementation
In 2022 IEEE 35th International System-on-Chip Conference (SOCC),
Pages 1 – 6. IEEE, 2022,
doi: https://doi.org/10.1109/SOCC56010.2022.9908130

9. Protecting Hardware-Intrinsic Features

The previous chapter was mainly concerned with the design of a secure system based on a PUF. In this, potential hardware-related attacks were disregarded. These are now the subject of this chapter. In Section 9.1, potential attacks on and countermeasures for an oscillator-based PUF, the Loop PUF, are developed and analyzed. As discussed in Chapter 7, it is not sufficient to implement the PUF securely. Also, the protection of the post-processing from a response to a secure key must be analyzed. Thus, Section 9.2 introduces a novel fault injection attack against the error correction in PUF designs. Even if the key is derived securely, secret information needs to be transferred on a chip. In particular, if this transfer happens via bus lines, a probing attack might be an issue. A corresponding countermeasure is introduced in Section 9.3.

9.1 Side-Channel Attacks and Counter Measures on a PUF Primitive

Side-channel attacks are one of the most severe attack vectors for PUFs deriving a secret from hardware-intrinsic features. These attacks can reveal some secret information by observing, e.g., the energy consumption of a chip or by observing EM radiations during the processing of secret information.

9.1.1 Methods for Attacking and Protecting Sign-Based Bit Quantization

The work in the following targets a Loop PUF as introduced in Section 4.2.4. Such a Loop PUF derives the secret from the difference of a frequency – expressed as a counter value – measured under a challenge **ch** and its bitwise inverse $\neg$**ch**. It has been shown before for RO PUFs that the frequency of an oscillation can be observed through a side-channel analysis [107], and it is no surprise that this is possible for the oscillator structure of a Loop PUF, too. However, for the classical Loop PUF, the situation is even worse: The same PUF is measured under the challenges and their corresponding complement one after the other, and an attacker can observe how the spectrum develops over time and, thus, find out the oscillation frequency of the Loop PUF under the different applied challenges.[1] Since the secret for a classical Loop PUF is derived from the

[1] In the basic Loop PUF design it is assumed that the challenge order is fixed and known to an attacker.

counters representing the oscillation frequencies, the attacker can derive in good approximation the secret information.

■ **Result 9.1.1** While it was expected that the secret could be extracted from a classical Loop PUF, we were able to show this in **[149]** with practical measurements: Out of 63 bits derived from the same PUF, we were able to extract 61 correctly. The errors in the prediction of the attack happened for cases where the frequencies of the PUF under **ch** and ¬**ch** are close. These are, however, cases where errors in the derived PUF response are likely and which need to be corrected by the device, too. The error correction and helper data are assumed to be public, and thus, the design can be considered to be wholly broken, with 61 out of 63 bits revealed. The results of our analysis, however, also show that the precision of a typical SCA, consisting of a time domain measurement with an oscilloscope followed by an FFT, does not suffice to resolve the LSB of the counter value counting the Loop PUF oscillations within a fixed time; A claim we were also able to substantiate theoretically.[2] ■

Please note that the analyzed scenario is a best-case scenario for the attacker. The attack difficulty is increased if more than one PUF runs on the device at the same time. However, the attack is expected to be still feasible in the more difficult setting, and the simplification is acceptable since the goal of the attack was to establish the ground for the development and test of a countermeasure.

Since, in this first scenario, only a single bit is derived from a pair of measurements, it is sufficient for protection to add some uncertainty so that an attacker no longer knows if the challenge or the complementary challenge was measured first. Based on this observation, an efficient countermeasure was developed:

■ **Result 9.1.2** If the noisy LSB of the counter value of a Loop PUF cannot be easily resolved, it can be taken as a random bit. We measure the Loop PUF under a first, otherwise unused challenge in order to generate such a bit. Depending on this random bit, the measurement order of **ch** and ¬**ch** for the next secret is either kept or inverted. Since the random bit is not known, the attacker can measure the distance between frequencies but not the secret sign of the difference. At the same time, the new LSB from a frequency measurement can be taken to protect the derivation of the next secret. By this form of temporal masking, the Loop PUF protects itself without the need for an additional TRNG. The experiments in **[149]** support this argumentation. ■

The attack so far was done with an oscilloscope with a sampling frequency of 1.25 GHz. However, it was not clear if it is still feasible if an attacker has much worse equipment in place. Such a scenario we analyzed in **[154]**. In that case, the attack was mounted remotely, e.g., on a multi-tenant FPGA. The attacker, in that case, can program their IP on the same FPGA on which the victim generates a secret with the PUF but has no direct access to take measurements. Related scenarios have been successfully analyzed for cryptographic algorithms.

The method to mount a remote SCA is to realize some measurement circuit, usually a Time-to-Digital Converter (TDC). This TDC in our work consists of a delay chain and a sequence of latches, both driven by the same reference clock. Within one period of the reference clock, the clock signal propagates through a certain number of elements in the delay chain. This chain is sampled using latches. The number of elements the signal propagates through depends on the individual delay of the gates but also on, e.g., the supply voltage. If another circuit's computation causes small voltage variations to the supply of the TDC, the number of traversed delay elements changes, enabling observation of such side-channel leakage.

■ **Result 9.1.3** With the concept of a TDC, it is possible to observe the frequency of a PUF. The

[2] Please note that novel mathematical transforms might be used to reveal the LSB, in particular, if the RO oscillates at a low frequency.

SNR with such a low-cost solution is, however, much worse so that several hundred repetitions are needed to predict the Loop PUF response with similar precision as it is possible with a single oscilloscope measurement. ■

9.1.2 Methods for Attacking and Protecting Amplitude-Based Quantization

While the temporal masking approach above protects the sign bit, it is still possible for an attacker to derive the absolute difference between frequencies. This affects all not purely sign-based quantization schemes introduced in Section 4.4: Given, e.g., the magnitude of a frequency difference but not the sign for equiprobable or equidistant quantization, only two intervals can be reached, reducing the entropy under given side-channel information to one bit. For the TMHD scheme, only a single bit is derived per frequency difference. This bit is revealed using – depending on the concrete implementation – the helper data and the information about the magnitude. The reason is that the quantization to a response bit value of zero or one, as well as the derived helper data, is based on the quantiles of the expected distribution of the analog responses. An attacker with access to the absolute frequency differences generated by the PUF can estimate this distribution and do a similar quantization step as the device itself. Also, for the Lehmer-Gray Encoding, knowing the magnitude reduces the possible orders when sorting the frequencies, resulting in a loss of entropy.

■ **Result 9.1.4** While several quantization techniques and PUFs are affected by the general results, the use case considered in the research is a combination of the TMHD scheme and a Loop PUF. In **[151]**, we have shown mathematically that such a magnitude-based attack is possible, reducing the entropy in the response after the TMHD scheme to zero. The feasibility of such an attack has been shown experimentally in the same publication, too. ■

There are two strategies to prevent the attacker from mounting an attack exploiting the magnitude of frequency differences in this case: The attack is prevented if (i) the attacker does not know in which order the challenges are applied to the Loop PUF or (ii) the attacker is not able to resolve different frequencies.

In order to prevent the attacker from knowing the challenge order, it is necessary to randomize them not only pairwise – as it has been done for the temporal masking – but in a way so that the sequences of all challenges become unpredictable. At the same time, it must be the goal to be lightweight and to reduce the randomness as much as possible. Two general approaches have been considered to reach this goal: First, we demonstrated that a variant of Fisher-Yates shuffling is an appropriate solution to the problem but requires – due to the need for rejection sampling – a comparably large amount of randomness. Second, we demonstrated that a simple method using an LFSR-based randomization strategy, which might seem a sufficient solution at first glance, is not secure.

■ **Result 9.1.5** In order to efficiently implement the randomization of the challenges, we suggested a permutation generator in **[151]** by inventing a form of Fisher-Yates shuffling: The set of challenges from which one has to choose reduces with each choice and does not always have a size, which is a power of 2. However, a source of randomness in hardware produces typically binary random numbers, which implies that the RNG output is always such a power of 2. Thus, rejection sampling is a reasonable approach to select one challenge after the other without replacement. However, to receive a completely unbiased distribution, a large number of rejections might be needed.[3]

As an alternative, we suggest choosing for each challenge a random number of a *fixed length*. If

[3] If e.g., 33 challenges remain in the set, a number from $[0,32]$ must be sampled. This requires 6 bits. However, there is a chance of close to 50% to sample with 6 bits a number in $[33,63]$, and these samples must be rejected if no bias shall be introduced.

the number of bits in the random number N_r is larger than the number of possible challenges N_{ch},[4] the probability that the same random number appears twice in N_{ch} samples reduces with every additional random bit. Sampling for all N_{ch} challenges *different* random numbers using rejection sampling and sorting the challenges in accordance with these numbers results in a random permutation. With this approach, we were able to show that, on average, approximately 577 random bits are needed for permuting 63 challenges. ■

While this result already protects the TMHD scheme, a more lightweight pseudo-random permutation would be reasonable for low-cost devices. Thus, the research in **[151]** analyzes a permutation using an LFSR of minimum size.

■ **Result 9.1.6** For the Loop PUF in combination with a Hadamard codeword and neglecting the all-zero codeword, there are $N_{ch} = 2^z - 1$ different challenges, with 2^z the number of stages of the Loop PUF. If these challenges shall be sorted in a pseudo-random manner, an LFSR of length z with primitive feedback polynomial is sufficient: In this case, the state of the LFSR produces all possible challenge indexes in a pseudo-random order. To make this pseudo-random permutation unpredictable, (i) random seeds, (ii) multi-bit shifts, (iii) a randomly permuted – masked – state, and (iv) different feedback polynomials can be used. While this is not perfectly secure, it would end up in close to 20 bits of entropy for an attacker to guess the correct order. If the attacker is not able to confirm the correct choice without guessing the complete key, this number might be sufficient for certain lightweight applications, since also for the attack on a Loop PUF, which is only protected by temporal masking, significant guessing and measurement effort is needed.

However, we developed in **[151]** another attack strategy on such a protection scheme: An attacker can interpret frequencies or frequency differences as symbols. Correlating the symbol sequences with the approx. 2^{20} different sequences produced by the LFSR reduces the possible choices for the sequence. As a result, an attack on such a lightweight permutation is feasible, and this method cannot be suggested for protection. ■

Since the effort of implementing the permutation of challenges for the Loop PUF is relatively high, the second option to prevent attacks on magnitude-based quantization, namely hiding the frequency difference from an attacker, was explored, too. This has led to the invention of the Interleaved Challenge Loop PUF (ICLooPUF) in **[150]**.

■ **Result 9.1.7** The ICLooPUF is derived from the Loop PUF based on the idea that a single period of oscillation is hardly resolvable by an attacker. Thus, we implemented an approach where the Loop PUF's response is generated by alternately observing one period of the oscillation under the challenge and one under the complementary challenge. Apparently, with this modification, it is no longer possible to compare the number of oscillations of the Loop PUF under a challenge and the corresponding complementary challenge within a fixed time, which would be needed to measure the frequencies and compare them. To solve this problem, we transformed the measurement apparatus into a time-domain measurement. I.e., the period length was approximated by oversampling and a counter counting the sample clock counts up for one period of the Loop PUF and down for the next period. This way, after several periods, the counter contains an approximation for the difference of the period lengths, which is then quantized. ■

Analyzing this idea in practice has shown that the approach is indeed feasible. However, significant oversampling is needed to reduce the risk of large quantization errors and wrong results, limiting the speed the Loop PUF can run with. Since the Loop PUF typically has many stages and has, thus, a relatively low frequency, this is no practical issue on the design side.

The implemented ICLooPUF was also theoretically and practically analyzed in **[150]**.

[4] Please remember that the number of challenges for a Loop PUF is limited to the challenge length due to the use of Hadamard codewords.

■ **Result 9.1.8** Theoretical and practical evaluations of the ICLooPUF have shown that the difference between two periods cannot be observed by an attacker when measuring in the time domain and analyzing the spectrum after an FFT in the frequency difference. In addition, the results have shown that the ICLooPUF in combination with the TMHD scheme has excellent performance w.r.t. reliability and indicate that the PUF quality in terms of unpredictability is sufficient, too. ■

The results shown in the research so far indicate that challenge permutation is a secure but costly protection method for SCA attacks on magnitude-based quantization schemes. The ICLooPUF, in contrast, seems a good starting point for a low-cost solution but is a modification of the PUF primitive and, thus, not generic. It should also be mentioned that novel yet unpublished results indicate that an attack exploiting the zero crossing of the up-down sample counter for the ICLooPUF might be possible and that for low-frequency Loop PUFs, for which significant oversampling with an oscilloscope is feasible, novel transforms might reveal exploitable information about the difference in the period length. Thus, further investigations and improvements to the ICLooPUF approach are still needed.

9.1.3 Impact on Other Oscillator Based PUFs

The research presented in this section was centered around the Loop PUF. However, some essential findings are also relevant for other oscillator-based PUFs and beyond. First, it is worth mentioning that the demonstrated attacks, including the novel remote SCA using a TDC, seem to be feasible not only when a Loop PUF is used but also for other oscillator-based PUFs.

Regarding the sign-based attacks and countermeasures, it is worth noting that different from the Loop PUF, most oscillator-based PUFs use frequency – or other – comparisons of spatially close but distinct primitives. Thus, the temporal protection might only be applicable to a counter used sequentially for different oscillator instances. While the counter in such a structure typically leaks most from a side-channel perspective, leakage of the oscillators themselves cannot be precluded. Nevertheless, it has been suggested in the past to shuffle the assignment from oscillator to counter in order to make SCA harder. For this purpose, the idea to use the LSB of a counter counting oscillator periods for protection might be useful.

With respect to magnitude-based quantization, in particular, the observations regarding challenge randomization are of relevance. If no localized measurements must be taken into consideration as an attack scenario, the same techniques used for the Loop PUF are applicable to randomize the order of measurements of different oscillators.

It is also worth highlighting that the apparent weakness of a Loop PUF, namely that an attacker can resolve different measurements over time and does not need localized measurements to resolve different oscillations, turns into a strength when it comes to side-channel protection: The results from the research show that randomization in the order of measurements seems to be more effective and better to implement than protections that must hide spatial dependencies.

9.1.4 Bibliographic Information

Results 9.1.1 and 9.1.2 discussed in this section have been presented in **[149]**, which appeared as:

Lars Tebelmann, Jean-Luc Danger, and Michael Pehl
Self-Secured PUF: Protecting the Loop PUF by Masking
In International Workshop on Constructive Side-Channel Analysis and Secure Design,
Pages 293 – 314. Springer, 2020
doi: `https://doi.org/10.1007/978-3-030-68773-1_14`

Result 9.1.3, showing the attack on a Loop PUF in a remote scenario was presented in **[154]**, which appeared as:

Lars Tebelmann, Moritz Wettermann, and Michael Pehl
On-Chip Side-Channel Analysis of the Loop PUF
Proceedings of the 2022 Workshop on Attacks and Solutions in Hardware Security
Pages 55 – 63, ACM, 2022
doi: `https://doi.org/10.1145/3560834.3563827`

Results 9.1.4, 9.1.5, and 9.1.6, demonstrating a first attack on magnitude-based quantization schemes and discussing countermeasures through challenge permutation for the Loop PUF, have been presented in **[151]**, which appeared as:

Lars Tebelmann, Ulrich Kühne, Jean-Luc Danger, and Michael Pehl
Analysis and Protection of the Two-Metric Helper Data Scheme
In International Workshop on Constructive Side-Channel Analysis and Secure Design Pages 279 -– 302, Springer, 2021
doi: `https://doi.org/10.1007/978-3-030-89915-8_13`

Results 9.1.7 and 9.1.8, introducing the ICLooPUF and demonstrating its quality, have been presented in **[150]**, which appeared as:

Lars Tebelmann, Jean-Luc Danger, and Michael Pehl
Interleaved Challenge Loop PUF: A Highly Side-Channel Protected Oscillator-Based PUF
IEEE Transactions on Circuits and Systems I: Regular Papers
Vol. 69(12), Pages 5121 -– 5134, IEEE, 2022
doi: `https://doi.org/10.1109/TCSI.2022.3208325`

9.2 A Fault Injection Attack on the PUF Postprocessing

The previous section has demonstrated the feasibility of and countermeasures against SCA attacks on PUF primitives. However, it is not sufficient to analyze the PUF only. As discussed in Section 7.2.2, side-channel attacks exist targeting the postprocessing of a PUF response to a secret key. This section now considers a FIA attack on this postprocessing, more precisely on the error correction decoder.

9.2.1 Proposed Method and Results

As a use case for the analysis regarding the feasibility of a FIA attack on the PUF postprocessing, a concatenation of a 7-Repetition and a $(n,k,d) = (127,64,21)$ BCH code was used, which is a representative of the most commonly used code construction for PUFs in literature. A Fuzzy Commitment Scheme was used as HDA. The repetition decoder takes a sequence of 7-bit wide code words computed from the PUF responses XOR the helper data and provides a single BCH codeword bit as an output. The BCH codeword bits are sequentially fed into the BCH decoder. To make the results independent from the PUF construction, the PUF was simulated for the experiments, which provides full control over the PUF induced errors and the PUF response.

The novel attack uses a two-step approach:

1. The error correction code is brought to its correction limits so that any additional error results in a wrong key.
2. A glitch is inserted into the design, resulting in an exploitable error, namely in the observation of whether the key is correct or not, which is assumed to be observable from using the key in some cryptographic algorithm.

The first step can be achieved directly by helper data manipulation as described in Section 7.1. For the second step, a glitch was introduced by an on-chip glitch generator on FPGA, which constitutes a best-case scenario for the attacker.

■ **Result 9.2.1** Our analysis has shown that inserting a glitch results in a setup time violation for the input bits, which must propagate from some source through the repetition decoder into the BCH decoder. As a consequence, a bit b_A, applied to the BCH decoder before the glitch, is processed twice, and bit b_B, which actually should be used, is skipped. This way, b_B is replaced by b_A. With the strategy of bringing the BCH code to its correction limits in the first step, this results in two different situations: If $b_B = b_A$, the codeword is corrected to the correct one; if $b_B \neq b_A$, the resulting codeword, and thus the key, is incorrect. An attacker can distinguish these two cases. In other words, the attacker gains the knowledge if two subsequent bits are equal. For an idealized attack, the attacker would reveal all dependencies and would be left with one bit of entropy, i.e., the choice of some reference bit being logically zero or one.

We have shown, however, that in our practical setting, the attacker cannot reveal all dependencies between codeword bits. This is the case, e.g., since, at some positions, the inserted glitch affects not only the data path. This results in the problem that the derived key is never correct when inserting a glitch at this position. In addition, the insertion of the glitch-caused error does not always succeed, so multiple repetitions of the attack are necessary to reveal the correct secret. Consequently, while the attack is still feasible, the attacker remains with higher effort and a higher uncertainty than theoretically possible. Different guessing strategies have been explored, utilizing also that an attacker might predict which bit positions always fail. It was found that, on average, for attacks on different FPGAs and a noise-free PUF, around nine out of sixty-four bits of entropy remain for an attacker under the best guessing strategy. ■

While the previous result already demonstrates that the attack is feasible, there are different shortcomings: First, the PUF is assumed to be noise-free; Second, the attack can be prevented when the helper data is hashed with the error-corrected secret to a key since, in this case, the key

will always be wrong if helper data is manipulated; Third, we rely on an on-chip glitch setup. Starting with the third shortcoming, an off-chip-generated glitch is likely much less precise, but more repetitions are expected to remedy the effect. Another experiment addressed the first and the second shortcomings.

■ **Result 9.2.2** The requirement for the attack is that the BCH decoder is on its error correction bound. However, the error correction is designed to correct all errors in the worst case but not much more. Thus, it might be possible to modify supply voltage and environmental conditions in a way so that the error correction is already at its bound. Within such a setting, no helper data manipulation is needed anymore, and consequently, hashing helper data would not hinder the attack.

The experiment, in this regard, manipulated the PUF so that, on average, the BCH decoder was on the error correction limit and considered a noisy PUF. The attack indeed performed significantly worse, and for some FPGA boards, no successful attack was possible. However, for most cases, it was still possible to mount a successful attack, although the remaining entropy was higher than for the noise-free case with helper data manipulations. ■

Since the attack is very powerful and challenging to prevent, it is worth exploring if existing SCA countermeasures – in particular, the codeword masking mentioned in Section 7.2 – are effective against this FIA attack.

■ **Result 9.2.3** Applying the described attack on implementation with codeword masking revealed that such a countermeasure can be effective if it is implemented correctly. However, depending on the implementation, the attack might get even more efficient for such an implementation so that no general conclusion can be drawn. Further, it can be expected that the attack is also effective for decoders other than the BCH decoder. However, highly parallelized implementations of error correction decoders might be less sensitive against this kind of FIA, but usually result in a higher area overhead. Further research is needed in order to confirm this expectation and to come up with optimized solutions against this kind of attack. ■

9.2.2 Bibliographic Information

This work **[131]** appeared as:

Jonas Ruchti, Michael Gruber, and Michael Pehl
When the Decoder has to Look Twice: Glitching a PUF Error Correction
IACR Transactions on Cryptographic Hardware and Embedded Systems,
2022(3), Pages 26 – 70
doi: `https://doi.org/10.46586/tches.v2022.i3.26-70`

9.3 Protection Against Probing Attacks

In the previous sections, attacks on PUF primitives and their postprocessing were shown. However, after a secret is derived on a chip, the secret needs to be transmitted to a cryptographic core, and other sensitive information needs to be sent over wires on the chip. A powerful attack to reveal such secret information is a probing attack. I.e., an attacker attaches a probe to a device in order to read out secret information as discussed in Section 5.3. One countermeasure is to detect the case if a probe is attached and to transmit secret information only if no such case is observed. In this, detector circuits must provide low false positive as well as false negative rates since too large false positive rates would prevent the circuit from operating correctly, which reduces manufacturing yield, and a too large false negative rate would open the door for probing attacks. Existing solutions frequently have the problem of requiring significant calibration effort to achieve a good trade-off.

9.3.1 Proposed Method and Results

The novel probing detector introduced in **[110]** significantly reduces this problem by applying differential measurements of oscillator structures. In this, the work focuses on the protection of bus lines, which are in the first step of equal length. The bus lines are included in the oscillator, and a probe attached to one of the bus lines is detected since it slows down the oscillation frequency. To measure the difference, two counters are instantiated, counting the oscillations, and the first counter reaching some reference value stops the measurement.

■ **Result 9.3.1** Since differential measurement between two identically designed oscillators is applied, variations due to temperature and global process variations cancel out to a large extent. The use of an oscillator causes the same line to be measured multiple times, making the detector more sensitive against small probes. Nevertheless, process variations and other effects have an impact so that the difference between the two counters counting a pair of oscillators is likely different from zero when counting for long enough. A so-called decision bound accounts for this effect. I.e., only if the counter difference is small enough, no alarm is raised. **[110]** has evaluated different decision bounds and has shown that for sufficiently long measurement time in terms of oscillations of the ROs, probes can be detected with high probability while keeping the false positive rate low. ■

The problem of detecting a probe gets more complicated if the lines of the bus have different lengths. This is, however, frequently the case in practical applications. To overcome this problem, **[111]** enlarges the concept by a calibration step.

■ **Result 9.3.2** Taking the decision bound, which provides the best trade-off in **[110]**, experiments first show that the case of bus lines with nominally the same length is indeed the case for which probes are the easiest to detect. They further show that for long bus lines with corresponding significant differences in length, no reliable detection is possible anymore with the initial approach. Thus, the distribution of counter differences for the probed and unprobed cases has been analyzed. The two cases can be modeled by close-to-Gaussian probability distributions, for which an optimal distinguisher can be found. Apparently, each oscillation frequency is influenced similarly by the additional capacitance introduced by an attached probe. Thus, the characterization allows for coming up with a fixed offset value from the expected counter difference to define when a bus line is considered to be unprobed and when it is considered to be probed. Only the expected counter difference value must be characterized, which is only at nominal conditions and without an artificial probe capacity attached. This is considered a feasible task in the manufacturing process and is a benefit when compared to other probing detectors that might require calibration with and without a (faked) probe in place in order to achieve very good results. ■

9.3.2 Bibliographic Information

This basic concept of the probing detector appeared in **[110]** as:

Seyed Hamidreza Moghadas and Michael Pehl
ROPAD: A Fully Digital Highly Predictive Ring Oscillator Probing Attempt Detector
In 2020 57th ACM/IEEE Design Automation Conference (DAC),
Pages 1 – 6, 2020
doi: https://doi.org/10.1109/DAC18072.2020.9218546

The corresponding extension to bus lines with non-equal lengths appeared in **[111]** as:

Seyed Hamidreza Moghadas, Michael Pehl, and Georg Sigl.
ROPAD+: Enhancing the Digital Ring Oscillator Probing Attempt Detector for Protecting Irregular Data Buses.
IEEE Transactions on Very Large Scale Integration (VLSI) Systems,
vol. 30, no. 11, pp. 1716-1727, 2022,
doi: https://doi.org/10.1109/TVLSI.2022.3191471

10. Statistical Evaluation of PUFs

The previous chapter has discussed side-channel leakage and physical attacks as a threat to the extraction of secrets from hardware-intrinsic features. However, if the PUF is not perfect, the responses become at least partly predictable already from a statistical analysis. It was discussed in Chapter 6 that several metrics exist to evaluate the quality of PUFs. So why is more research needed? Overall, the state of the art in analyzing the predictability of PUF-derived secrets reveals two gaps: First, the evaluation usually focuses on binary PUF responses. This is addressed in Section 10.1. Second, estimates of the PUF entropy require some improvements. A contribution is provided in Section 10.2. In addition, this chapter introduces in Section 10.3 a novel PUF design and shows (i) how the quality of such a design can be considered already at design time and (ii) discusses methods for a more meaningful estimation of bit error rates compared to the usual state of the art in PUF literature.

10.1 Bias Estimation for Higher-Order Alphabet PUFs

Frequently, for PUFs, analog measurement results are available that are quantized in a subsequent step. The most common quantization techniques provide a mapping to a single bit by comparing differences between measurement results. However, more entropy per area can be gained by quantizing the analog responses into higher-order alphabet symbols using, e.g., the equiprobable or equidistant quantization discussed in Section 4.4. While these methods are known, state-of-the-art evaluation methods for PUFs frequently fail to provide appropriate test methods and focus instead on binary responses.

Seizing on the idea of different dimensions of PUF evaluation shown in Fig. 6.1 on page Page 45 bias must be considered over chips, positions on a chip, and challenges, and correlations must be considered between positions on a chip and over challenges. For the correlation metrics like Spatial Autocorrelation or Fisher-Yates Test in Section 6.3.3, straightforward approaches exist to apply them to higher-order alphabet PUFs. However, the bias metrics, approximating the expected PUF response by the relative frequency in a binary string, cannot directly be adapted: It is intuitive to call a higher-order response biased if some symbols appear with higher frequency than others. The expectation value of a higher-order response does, however, not necessarily differ from the ideal value if the symbols are chosen from such a suboptimal distribution.

10.1.1 Proposed Method and Results

To overcome the difficulty with testing the bias of higher-order alphabet PUFs, three methods have been suggested and investigated in **[44]**: Pearson's Chi-square Test, a test based on multinomial confidence intervals, and a test defining acceptable intervals. All of these tests are statistically sound tests in the sense that they test if hypothetical properties are fulfilled and that p-values for the test are computed, providing confidence in the result.

■ **Result 10.1.1** Pearson's Chi-square Test compares the observed distribution of symbols with an expected one. The test has been adapted for the case of PUF testing so that the expected probability for every symbol is not necessarily fixed but has to be within a certain range. This approach allows for testing realistic PUFs, for which a particular bias might be acceptable even if such a bias strictly requires postprocessing to compensate for it. To provide reasonable results, for Pearson's Chi-square Test, the test should be designed so that the expectation for every symbol to appear is sufficiently large, implying an overall huge number of samples if the expected probabilities for some symbols are low. The main drawback of Pearson's Chi-square Test is that it does not provide insights into why the test fails. In addition, if the null hypothesis of a PUF following the expected distribution cannot be rejected, no clear statement of the PUF quality can be made.

The second test, based on multinomial confidence intervals, computes a confidence interval per symbol, providing insights regarding what the actual distribution of the symbols might be. The test, thus, provides confidence if a PUF has good quality: Good quality can be expected if the required probability for each symbol is within the confidence interval and the confidence intervals are small enough. However, if the interval size is not small, the actual distribution of symbols can also be far off the intended distribution. An interesting approach of the computed confidence interval is, however, that they can be used to provide a lower bound for the min-entropy of a response with a certain confidence.

The third test, which defines acceptable intervals, requires uncorrelated data. Under this assumption, similar to the approach in **[167]** for binary responses, the null hypothesis that the PUF is of too low quality can be formulated. This corresponds to the hypothesis that the probability of occurrence for any symbol is larger or smaller than a specific value. The test operates on each symbol individually and ensures – if it is rejected – a good PUF quality with high confidence. The test is, in particular, useful if a certain bias can be compensated by post-processing in a PUF design and high confidence is needed that the requirements for this post-processing are not violated. ■

The tests described in this section are, in principle, sufficient to test any higher-order alphabet PUFs regarding their bias. Experimental results in **[44]** reveal their strength and limitations.

■ **Result 10.1.2** To start with the limitations, two points shall be mentioned: First, bias tests work reliably only if correlations between bits can be excluded. Thus, as a first step, a correlation test should be applied when testing any PUFs. Second, for HOA PUFs, even more samples are needed when compared to PUFs with binary response. As a consequence, testing HOA PUFs is generally more expensive, and this problem cannot be overcome with the introduced methods. In particular, high confidence in the test results of a single chip might be hard to achieve since the number of PUFs on a chip is bounded by design. Nevertheless, the results of the tests have shown that the suggested metrics can detect weaknesses where other metrics fail and are well suited for testing PUFs. ■

10.1.2 Bibliographic Information

The results in this section were published in **[44]**, which appeared as:

Christoph Frisch and Michael Pehl
Beware of the Bias – Statistical Performance Evaluation of Higher-Order Alphabet PUFs
2022 Design, Automation & Test in Europe Conference & Exhibition (DATE),
Pages 1005 – 1010, IEEE, 2022
doi: https://doi.org/10.23919/DATE54114.2022.9774554

10.2 Entropy Estimation for PUFs

While the statistical evaluation of a PUF with bias and correlation metrics is a reasonable tool to analyze and guarantee the quality, measuring the entropy in the PUF response provides a more global view of the PUF quality. Furthermore, in the end, the entropy in the PUF-derived key is a strict constraint when designing a system. Thus, entropy metrics for PUFs are discussed in this section. In this regard, it is worth noting that the term "entropy" in the PUF context refers to the exploitable entropy resulting from process variations. Variations due to noise effects are either not considered or do not provide a positive contribution to the PUF entropy defined this way.

10.2.1 Improving Compression-Based Estimates

In Section 10.1, it was mentioned that based on the statistical evaluation with multinomial confidence intervals, a min-entropy could be computed. However, this does not take into account correlated PUF responses. Thus, the entropy estimation in this section is based on the CTW approach introduced to the PUF domain in [69]. The approach constructs a context tree based on a sequence of symbols and uses it to derive an upper bound for the joint entropy based on this. While it was demonstrated that the approach produces good results in the evaluation of PUFs, it has one significant drawback: It considers the data a stream and constructs the context tree based on this stream. If two symbols are not in the same context, correlations between them might not influence the entropy estimate.

■ **Result 10.2.1** Many PUFs are arranged as an array. In these cases, correlations between spatially neighboured PUF cells are more likely since physical effects cause them to appear. To cover such cases, spatial contexts are defined in **[124]** and it has been shown that the asymptotic properties of the CTW algorithm are not affected by this modification. The approach was tested with different actually measured PUF data for which the results show that this *Spatial CTW* provides a tighter bound and converges faster for the case of a PUF. In addition, using different shapes for the spatial context allows for identifying potential weaknesses in the PUF.

The developed approach is also practical for HOA PUFs, where the main difference is that a non-binary tree is built to compute the probabilities needed for entropy estimation. ■

It is worth noting that – while we achieved a tighter bound – other modern compression algorithms might be used to replace CTW. Nevertheless, the results indicate that it is reasonable to consider spatial dependencies in the data structure for such algorithms, too. It is also important to keep in mind that any lossless compression algorithm provides only an upper bound for entropy. As such, it cannot prove a good quality of a PUF but can rather show if the entropy provided by a PUF is too low for some application scenario.

10.2.2 Estimation of Entropy in a Key

While the evaluation methods in this section focused so far on the quality of the PUF response, the focus is shifted in the following towards the key. Apparently, the entropy in a PUF-derived secret cannot be larger than the entropy by the PUF itself. However, for the key, the attacker has some additional knowledge in the form of helper data. This is, instead of estimating the entropy of the PUF response $H(\mathbf{p})$, the conditional entropy $H(\mathbf{p}|\mathbf{w})$ of the PUF response given the helper data must be computed. The result corresponds to the uncertainty regarding the knowledge of a secret for an attacker.[1]

As it has been discussed in Section 6.3.4, previous work in [35, 164] has introduced the basic mathematical concept for estimating the average conditional min-entropy for this case: These concepts exploit that not every response appears with the same probability. With the side knowledge

[1] When hashing the secret to a key, entropy might be further reduced.

of the helper data, this means that certain codewords are much more likely than others. Exploiting this knowledge, the central formula for the average conditional min-entropy is derived in [35] such that for every possible helper data sequence used to reconstruct a codeword, the probability of the most likely codeword is summed up, and the result is divided by the number of codewords and finally logarithmized to compute the entropy.

For realistic code length, it is not possible to iterate over all possible helper data sequences and to sum them up. However, for PUFs, linear codes are typically used. For these codes and under the assumption of equally likely codewords[2] the number of operations to compute the average conditional min-entropy for an (n,k,d) code can be reduced to 2^{n-k}. This is still too much for practical codes, so that for realistic code-length improvements are needed. [35] has reduced the number of operations by grouping all PUF response sequences that result in the same probability. This is possible with reasonable effort for the cases of iid biased responses or equal correlation between all bits, which are the cases considered in [35]. However, for practical PUF responses with different biases per position, every response has its probability. So grouping is not possible anymore. For this case, **[164]** suggested an approximation.

However, even with these simplifications, the computation effort is too high for analyzing practical codes. The challenges are (i) to compute the different probabilities needed and (ii) to derive the conditional average min-entropy based on this. Thus, [163] introduces a Response Mass Function (RMF). An RMF is defined as a discrete distribution that provides for every expected probability of a response the relative frequency of occurrence of precisely this probability. The RMF of the complete PUF is constructed through convolution from RMFs of sub-groups of PUF responses. If all probabilities appear precisely once, still the number of different probabilities would be too large for practical computation of the average conditional min-entropy. However, instead of assigning every probability of occurrence in the RMF an individual relative frequency, multiple probabilities can be conflated into a single bar of a histogram representing probabilities of responses within a certain range. The resulting limited number of bars of the histogram can be used to compute the average conditional min-entropy of the PUF given the helper data.

■ **Result 10.2.2** Applying this approach in **[46]** to codes of practical length and using actual and synthetic PUF data provides not only confidence in the applicability of the method. It also provides insights into how codes shall be designed for PUFs. Overall, the findings can be summarized as follows: Unsurprisingly, less helper data are always beneficial. This means that codes with higher rates are preferable. Also, shorter codes of the same rate leak less. The practical results also show that for the considered PUF in **[46]**, which has an approximated min-entropy in the PUF response of more than 0.7 bits per response bit, for the best code rate achievable with a linear block code – a case that is defined through the Griesmer bound and serves as a reference – the entropy per bit when considering error correction is slightly above 0.3 bits per message bit. The frequently used concatenation of a 7-repetition and a $(n,k,d) = (127,64,21)$ BCH code results in an average conditional min-entropy of slightly above 0.1 bits per message bit.

Additional experiments with synthetic data allowed for the insertion of a controlled correlation between symbols. For this case, a hypothetical RO PUF with overlapping comparison[3] was considered, a case that appeared in literature and is known to be – without further protection mechanism – vulnerable against side-channel attacks and to contain correlations. With the novel estimation approach, computation of the average conditional min-entropy was possible. The result illustrates that the loss in entropy with such an overlapping comparison is already for small groups

[2]This assumption can be assumed to be fulfilled if the codeword is chosen at random, e.g., in a Fuzzy Commitment or Code-Offset Construction.

[3]For four ROs frequencies f_A, f_B, f_C, and f_D, e.g., an overlapping comparison would derive three bits from comparing the tuples $(f_A, f_B), (f_B, f_C), (f_C, f_D)$ while a non-overlapping comparison would derive two bits from the comparisons $(f_A, f_B), (f_C, f_D)$

of ROs so significant that a non-overlapping comparison provides more bits of entropy, although fewer response bits are produced. As a consequence, this research was able to demonstrate that an overlapping comparison is not only more vulnerable to SCA but also less effective in terms of entropy. ∎

10.2.3 Bibliographic Information

Result 10.2.1 was presented in **[124]**, which appeared as:

Michael Pehl, Tobias Tretschok, Daniel Becker, and Vincent Immler
Spatial Context Tree Weighting for Physical Unclonable Function
In 2020 European Conference on Circuit Theory and Design (ECCTD),
Pages 1 – 4, IEEE, 2020
doi: `https://doi.org/10.1109/ECCTD49232.2020.9218325`

Result 10.2.2 was presented in **[46]**, which appeared as:

Christoph Frisch, Florian Wilde, Thomas Holzner, and Michael Pehl
A Practical Approach to Estimate the Min-Entropy in PUFs
Journal of Hardware and Systems Security, 2023
`https://doi.org/10.1007/s41635-023-00139-x`

10.3 Design and Evaluation of a Novel PUF Primitive

While the chapter has focused so far on the evaluation of the unpredictability of a PUF, the last section, as well as the machine learning attack in Section 8.2, have also demonstrated that the error correction is relevant from a security perspective. From a high-level perspective, less error correction is always beneficial, not only for the reduction of area overhead but also for reaching a higher security level with fewer PUF responses and helper data. This can be reached if the error probability of the PUF is reduced.

10.3.1 Proposed Method and Results

A reduction of the error probability of a PUF response can be reached best by a design optimization of the measurement circuit of a PUF in the analog domain. The proposal in **[128]** targets this goal by introducing a resistor-based PUF. Integrated resistors on a device act as the source of randomness and are measured via the voltage drop when a well-defined current is applied. An analog compensation circuit removes temperature effects. By comparing pairs of resistors, effects like variations of the supply voltage or humidity-caused strain are canceled. Since the compared resistors are placed in spatially distinct areas on the chip and might see different global variations, an additional trimming of the currents flowing through the PUF elements is used to remove such effects. The circuit has been evaluated through exhaustive simulations, constituting results from 1000 chips with 200 PUF responses derived per chip from 200 voltage differences as the analog PUF responses. The characterization was done over a temperature range from $-40\,^{\circ}\mathrm{C}$ to $110\,^{\circ}\mathrm{C}$. It can be expected that the analog simulation models the actual behavior on the chip – despite correlations – well.

■ **Result 10.3.1** The analysis shows that the noise-free raw differences of analog voltage measurements are well modeled by a Gaussian distribution with a mean close to zero. This is the expected result for a good PUF regarding the analog responses. Also, applying classical performance metrics for unpredictability to the quantized PUF response demonstrates excellent PUF behavior. These results were confirmed also by applying improved metrics defined in **[167]**. However, it has to be noted that this result can only act as a first indicator for the actual PUF quality since correlations are not analyzed due to the limitations in the simulation, and an actually manufactured PUF might reveal unexpected effects reducing the quality. ■

With the unpredictability of the PUF confirmed, in addition, the reliability of the PUF was analyzed. Furthermore, the efficiency in terms of the number of bits needed to derive a secret key was estimated. For this estimate, a concatenation of repetition and $(n,k,d) = (127,64,21)$ BCH code was used, and a 128-bit key was targeted to be comparable to other research results.

■ **Result 10.3.2** To estimate the error probability of the PUF, a more systematic approach was taken than usually provided in the PUF literature. We start with the assumption that a specific yield must be achieved during the manufacturing of the chip. I.e., the error correction must be able to correct the error on a high percentage – here 99.73% which corresponds to a 3-sigma design – of chips down to a key or frame error probability of at most 10^{-6}. It follows that the BER must be predicted, which is met in the worst-case corner for the targeted percentage of chips. The error correction must then be designed to correct at least this error rate.

To come up with such a design, the temperature shift and the random noise must be considered. The temperature shift is a non-probabilistic effect, for which the worst case must be determined, but it does not have the same effect on every pair of resistors, which makes it challenging to model. Two approaches are suggested for this.

1. A *worst case model* is derived from the simulations by observing the maximum shift of the measured voltage differences, and it is assumed that – different from reality – all responses

are negatively impacted by this effect.[4] With this pessimistic model, a BER of nearly 18% was reached, which would make the PUF less effective than the state of the art.

2. A more realistic bit-error rate is derived from an *empirical model*. In that case, the distribution of the BER over chips at the worst-case corner is measured and modeled by a beta distribution. From this distribution and upper bound for the BER of a defined percentage of chips – e.g., 99.73% – can be concluded. With this model, a BER of approx. 11% was predicted, indicating that the PUF is more reliable than other state-of-the-art PUFs.

Since the predicted error rate is still relatively high, we suggest in **[128]** also a digital post-processing strategy: The mean temperature shift for some reference temperatures is determined at design time. When the device is measured at run time, a temperature-dependent value is added or subtracted to the observed (digitized) analog voltage difference before deriving the secret bit. With this compensation, the estimated error probability under the empirical model is only approx. 5% making error correction much more efficient. ∎

10.3.2 Bibliographic Information

The results in this section were published in **[128]**, which appeared as:

Carl Riehm, Christoph Frisch, Florin Burcea, Matthias Hiller, Michael Pehl, and Ralf Brederlow
Structured Design and Evaluation of a Resistor-Based PUF Robust Against PVT-Variations
2023 26th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS),
Pages 93 – 98, IEEE, 2023
doi: https://doi.org/10.1109/DDECS57882.2023.10139352

[4] In reality, the drift in voltages improves the reliability of some responses and decreases the reliability of others.

11. Conclusion and Outlook

In the last decade, significant progress has been made in exploiting hardware-intrinsic features. The main focus of research has been in this regard on silicon-based PUFs, for which a long list of possible implementations exists. This collection has provided a summary of the state of the art. It has been shown that the main applications exploiting hardware-intrinsic features are (i) secure key storage, (ii) authentication, identification, and fingerprinting, and (iii) tamper detection mechanisms. The techniques used in this regard still require refinement. In addition, they have to be continuously tested for security against novel attack strategies and require careful evaluation. Therefore, this collection contributed to these three areas.

The novel protocol introduced as a first contribution in this collection demonstrated how PUFs could be included in current designs and explored which strategies for a PUF design might be most efficient. The limitation if a Multi Challenge PUF is used in such a design was demonstrated by developing a novel machine-learning attack on PUF-based key storage. From the results, it can be concluded that Multi Challenge PUFs can be used for key storage only for a very limited number of keys without bearing the risk of getting vulnerable against machine learning. Nevertheless, modern Multi Challenge PUF designs might still allow for storing several hundred keys with a single PUF. This becomes, in particular, true if efficient error correction is implemented that requires only a low number of helper data and PUF bits to derive a key. The feasibility of such an efficient error correction scheme, namely the efficient implementation of a Polar decoder, was therefore also shown and discussed in this collection.

If hardware-intrinsic features are exploited to store a secret key, it is of great importance to ensure that this secret cannot be read out with hardware-related attacks. The reason is that hardware-intrinsic features are usually exploited in the context of lightweight applications for which physical access through an attacker cannot be excluded. Therefore, this collection also discussed hardware-related attacks on the PUF primitive and the error correction of a PUF. The focus regarding PUF primitives was on oscillator-based PUF primitives and, in particular, the Loop PUF. The research in this direction, however, needs to be continued: Several new and well-established PUF primitives are not yet researched regarding their vulnerability against SCA. In addition, FIA on such primitives is another open topic to be analyzed.

Regarding the postprocessing of the PUF, a novel FIA weakness was shown in this collection. This is a significant contribution since it emphasizes considering such attacks in the design of PUF-based secure key storage. Nevertheless, additional research is needed to better understand which decoder constructions are, in particular, insensitive against FIA and also against SCA attacks. Findings in this regard might provide insights not only relevant for PUFs but also, e.g., for code-based post-quantum cryptography.

In the context of attacks, also countermeasures against invasive attacks have been discussed. A new, promising probing detector was suggested. This research might also become of relevance for future invasive attacks on integrated circuits: A current trend is to implement complex systems through chiplets. Between the integrated chiplets, communication channels must be established over which sensitive data might be transmitted. It is likely not reasonable to encrypt every transmission between integrated chiplets so that other protection mechanisms, e.g., probing detection strategies, are needed to counter such an attack.

In the last contribution chapter of this collection, improvements towards a consistent evaluation of PUFs were proposed. These advances bridged gaps in the state of the art regarding considering higher-order alphabet responses from PUFs, spatial dependencies, and entropy estimations for a key. Although a standard for evaluating PUFs exists as discussed in state of the art in Section 6.3.6, this standard does not define a concise but complete test strategy. Developing such a test strategy is part of the ongoing research.[1] It is such a tool and a complete and reliable evaluation strategy that is a prerequisite for even wider usage of hardware-intrinsic features in secure systems and ultimately for certification of PUFs as the root of trust.

Besides analyzing the quality regarding unpredictability and reliability, the last section also discussed a novel PUF primitive based on resistors. This is an example of a systematic approach for exploiting hardware-intrinsic features with a novel technology. A similar analysis, but also the development of dedicated error correction strategies, will be needed for other novel PUF structures, e.g., when integrating memristor-based PUFs into secure systems.[2]

Overall, while different research questions remain for the future, this collection provided a snapshot of the state of the art at the time of its composition. The presented contributions provide significant insights and improvements regarding the design, hardware protection, and security analysis of PUFs and their postprocessing, as tools to derive secrets from hardware-intrinsic features.

[1] At the time of the compilation of this collection, the first version of such a strategy was under preparation for publication.

[2] The design and analysis of a BFO-based PUF co-authored by the author was under review at the time of the composition of this collection.

List of Acronymes

AEAD Authenticated Encryption with Associated Data. 63, 64

BCH Bose–Chaudhuri–Hocquenghem. 37, 58, 59, 68, 77, 78, 85, 87
BER Bit Error Rate. 46, 52, 87, 88
BFO $BiFeO_3$. 27, 90

C-IBS Complementary Index-Based Syndrome Coding. 36
COSO COherent Sampling ring Oscillator. 56
COTS Commodity-Of-The-Shelf. 25
CPA Correlation Power Analysis. 58
CTW Context Tree Weighting. 50, 51, 84

DPA Differential Power Analysis. 58, 59
DSC Differential Sequence Coding. 36

ECC Error Correcting Code. 33
EM electromagnetic. 55, 56, 58, 71

FeRAM Ferroelectric RAM. 31
FF Flip Flop. 57
FFT Fast Fourier Transform. 55, 72, 75
FIA Fault Injection Analysis. 14, 16, 53, 55, 59, 77, 78, 89, 90
FIB Focused Ion Beam. 45, 54, 57

HD Hamming Distance. 46, 48, 49, 58, 60
HDA Helper Data Algorithm. 32–36, 53, 57, 60, 68, 77
HOA Higher-Order Alphabet. 49, 55, 82, 84
HRS High Resistive State. 27, 28
HW Hamming Weight. 46, 47

IBS Index-Based Syndrome Coding. 36

ICLooPUF Interleaved Challenge Loop PUF. 74–76
iid independent and identically distributed. 44, 48–50, 52, 53, 65, 85

LFSR Linear Feedback Shift Register. 26, 74, 93
LLR Log-Likelihood Ratio. 68
LR Linear Regression. 57
LRS Low Resistive State. 27, 28
LSB Least Significant Bit. 56, 72, 75
LVP Laser Voltage Probing. 55, 56

ML Machine Learning. 59, 60, 63
MSB Most Significant Bit. 56

NVM Non-Volatile Memory. 27, 31–33, 54

PCA Principal Component Analysis. 52
PCM Phase Change Memory. 27
POK Physically Obfuscated Key. 23
PRNG Pseudorandom Number Generator. 57
PUF Physical Unclonable Function. 13–17, 19, 23–60, 63–66, 68, 71–79, 81, 82, 84, 85, 87–90, 92, 93

RMF Response Mass Function. 85
RNG Random Number Generator. 57, 59, 73
RO Ring Oscillator. 47, 55, 56, 59, 71, 72, 79, 85, 86
RO PUF Ring Oscillator PUF. 23–25, 46, 47
RRAM Resistive RAM. 27, 31
RS Reed-Solomon. 58

SCA Side-Channel Analysis. 14, 16, 27, 29, 53–60, 72, 75, 77, 78, 86, 89, 90
SNN Siamese Neural Network. 65, 66
SNR Signal-to-Noise Ratio. 26, 57, 73
SPA Simple Power Analysis. 58
STT-MRAM Spin-Transfer Torque Magnetoresistive RAM. 27

TDC Time-to-Digital Converter. 72, 75
TERO Transient Effect Ring Oscillator. 56, 59
TFF Toggle Flip Flop. 56
TMHD Two Metric Helper Data Scheme. 29, 73–75
TRNG True Random Number Generator. 17, 27, 32, 33, 39, 43, 44, 52, 53, 55–57, 59, 72
TTP Trusted Third Party. 41

List of Symbols

Please note that the literature used as contribution in Part III uses in parts different symbols.

$\mathbf{c}$	Codeword
$\mathbf{c}'$	Noisy version of $\mathbf{c}$
ch	Challenge bit
$\mathbf{ch}$	Challenge
d	Minimum distance of distinct codewords
$\neg\mathbf{ch}$	Bit-wise complement of $\mathbf{ch}$
$\mathbf{e}$	Error vector
$\mathbf{w}$	Helper data vector
k	Dimenion of a code
$\mathbf{k}$	Secret key
$\mathbf{k}'$	Reconstructed key from noisy data
L	Outputsequence of an LFSR
n	Length of a code
N_{ch}	Number of bits in $\mathbf{ch}$
N_p	Number of PUFs on a chip
N_r	Number of bits in $\mathbf{r}$
N_x	Number of chips
p	Response bit of PUF
$\mathbf{p}$	Response vector of PUF
$\mathbf{p}'$	Noisy PUF response vector
$\mathbf{p}_{raw}$	Analog PUF response
$\mathbf{r}$	Random number vector
$\mathbf{s}$	Secret vector
$\mathbf{s}'$	Reconstructed secret from noisy data
T	Time

Bibliography

In this Bibliography, publications with a contribution by the author are indicated in bold. Publications [45, 140, 80, 149, 154, 151, 150, 131, 110, 111, 124, 44, 46, 128] are the publications discussed in the contributions of this collection.

[1] ISO/IEC 20897-2. Information Security, Cybersecurity and Privacy Protection - Physically Unclonable Functions - Part 2: Test and Evaluation Methods, 2021.

[2] Isaac Abraham. The Case for Rejecting the Memristor as a Fundamental Circuit Element. *Scientific Reports*, 8(1):10972, Jul 2018.

[3] Anita Aghaie and Amir Moradi. TI-PUF: Toward Side-Channel Resistant Physical Unclonable Functions. *IEEE Transactions on Information Forensics and Security*, 15:3470–3481, 2020.

[4] **Md Toufiq Hasan Anik, Jean-Luc Danger, Omar Diankha, Mohammad Ebrahimabadi, Christoph Frisch, Sylvain Guilley, Naghmeh Karimi, Michael Pehl, and Sofiane Takarabt. Testing and Reliability Enhancement of Security Primitives: Methodology and Experimental Validation.** ***Microelectronics Reliability*****, 147:115055, 2023.**

[5] **Toufiq Hasan Anik, Jean-Luc Danger, Omar Diankha, Mohammad Ebrahimabadi, Christoph Frisch, Sylvain Guilley, Naghmeh Karimi, Michael Pehl, and Sofiane Takarabt. Testing and Reliability Enhancement of Security Primitives. In** ***2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*****, pages 1–8. IEEE, 2021.**

[6] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Berk Sunar, and Pim Tuyls. Memory Leakage-Resilient Encryption Based on Physically Unclonable Functions. In Ahmad-Reza Sadeghi and David Naccache, editors, *Towards Hardware-Intrinsic Security: Foundations and Practice*, pages 135–164, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[7] Daniel Arumi, Mireia Bargallo Gonzalez, and Francesca Campabadal. RRAM Serial Configuration for the Generation of Random Bits. *Microelectronic Engineering*, 178(C):76–79, June 2017.

[8] **Daniel Arumí, Salvador Manich, Rosa Rodríguez-Montañés, and Michael Pehl. RRAM Based Random Bit Generation for Hardware Security Applications. In *2016 Conference on Design of Circuits and Integrated Systems (DCIS)*, pages 1–6, Nov 2016.**

[9] Leonid Azriel and Shahar Kvatinsky. Towards a Memristive Hardware Secure Hash Function (MemHash). In *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 51–55, 2017.

[10] D. W. Bauder. An Anti-Counterfeiting Concept for Currency Systems. Technical Report PTK-11990, Albuquerque: Sandia National Labs, 1983.

[11] Pierre Bayon, Lilian Bossuet, Alain Aubert, Viktor Fischer, François Poucheret, Bruno Robisson, and Philippe Maurine. Contactless Electromagnetic Active Attack on Ring Oscillator Based True Random Number Generator. In Werner Schindler and Sorin A. Huss, editors, *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*, volume 7275 of *Lecture Notes in Computer Science*, pages 151–166. Springer, 2012.

[12] Georg T. Becker. The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems – CHES 2015*, pages 535–555, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[13] Georg T Becker, Alexander Wild, and Tim Güneysu. Security Analysis of Index-Based Syndrome Coding for PUF-Based Key Generation. In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 20–25. IEEE, 2015.

[14] Lilian Bossuet, Xuan Thuy Ngo, Zouha Cherif, and Viktor Fischer. A PUF Based on a Transient Effect Ring Oscillator and Insensitive to Locking Phenomenon. *IEEE Transactions on Emerging Topics in Computing*, 2(1):30–36, March 2014.

[15] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[16] Christina Brzuska, Marc Fischlin, Heike Schröder, and Stefan Katzenbeisser. Physically Uncloneable Functions in the Universal Composition Framework. In Phillip Rogaway, editor, *Advances in Cryptology (CRYPTO)*, volume 6841 of *LNCS*, pages 51–70. Springer Berlin / Heidelberg, 2011.

[17] Geoffrey W. Burr, Matthew J. Breitwisch, Michele Franceschini, Davide Garetto, Kailash Gopalakrishnan, Bryan Jackson, Bülent Kurdi, Chung Lam, Luis A. Lastras, Alvaro Padilla, Bipin Rajendran, Simone Raoux, and Rohit S. Shenoy. Phase Change Memory Technology. *Journal of Vacuum Science & Technology B*, 28(2):223–262, 03 2010.

[18] Suresh N. Chari, Vincenzo V. Diluoffo, Paul A. Karger, Elaine R. Palmer, Tal Rabin, Josyula R. Rao, Pankaj Rohotgi, Helmut Scherzer, Michael Steiner, and David C. Toll.

Designing a Side Channel Resistant Random Number Generator. In Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny, editors, *Smart Card Research and Advanced Application*, pages 49–64, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[19] Durba Chatterjee, Harishma Boyapally, Sikhar Patranabis, Urbi Chatterjee, Aritra Hazra, and Debdeep Mukhopadhyay. Physically Related Functions: Exploiting Related Inputs of PUFs for Authenticated-Key Exchange. *IEEE Transactions on Information Forensics and Security*, 17:3847–3862, 2022.

[20] Wenjie Che, Jim Plusquellic, and Swarup Bhunia. A Non-Volatile Memory Based Physically Unclonable Function without Helper Data. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 148–153, 2014.

[21] Bin Chen, Tanya Ignatenko, Frans M. J. Willems, Roel Maes, Erik van der Sluis, and Georgios Selimis. A Robust SRAM-PUF Key Generation Scheme Based on Polar Codes. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6, 2017.

[22] Chun. Chen, Raymond N. J. Veldhuis, Tom A. M. Kevenaar, and Anton H. M. Akkermans. Multi-Bits Biometric String Generation Based on the Likelihood Ratio. In *2007 First IEEE International Conference on Biometrics: Theory, Applications, and Systems*, pages 1–6, 2007.

[23] Li-Wei Chen, Ziang Chen, Werner Schindler, Xianyue Zhao, Heidemarie Schmidt, Nan Du, and Ilia Polian. On Side-Channel Analysis of Memristive Cryptographic Circuits. *IEEE Transactions on Information Forensics and Security*, 18:463–476, 2023.

[24] Pai-Yu Chen, Runchen Fang, Rui Liu, Chaitali Chakrabarti, Yu Cao, and Shimeng Yu. Exploiting Resistive Cross-Point Array for Compact Design of Physical Unclonable Function. In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 26–31, 2015.

[25] Zouha Cherif, Jean-Luc Danger, Sylvain Guilley, and Lilian Bossuet. An Easy-to-Design PUF Based on a Single Oscillator: The Loop PUF. In *2012 15th Euromicro Conference on Digital System Design*, pages 156–162, Sep. 2012.

[26] Abdelkarim Cherkaoui, Viktor Fischer, Alain Aubert, and Laurent Fesquet. A Self-Timed Ring Based True Random Number Generator. In *19th IEEE International Symposium on Asynchronous Circuits and Systems, ASYNC 2013, Santa Monica, CA, USA, May 19-22, 2013*, pages 99–106. IEEE Computer Society, 2013.

[27] Leon Chua. Memristor – The Missing Circuit Element. *IEEE Transactions on Circuit Theory*, 18(5):507–519, 1971.

[28] Jianwei Dai and Lei Wang. A Study of Side-Channel Effects in Reliability-Enhancing Techniques. In *2009 24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pages 236–244, 2009.

[29] **Jean-Luc Danger, Sylvain Guilley, Michael Pehl, Sophiane Senni, and Youssef Souissi. Highly Reliable PUFs for Embedded Systems, Protected Against Tampering. In *International Conference on Industrial Networks and Intelligent Systems*, pages 167–184. Springer, 2021.**

[30] Jean-Luc Danger, Sylvain Guilley, and Alexander 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)*, pages 184–188, 2019.

[31] Jayita Das, Kevin Scott, Srinath Rajaram, Drew Burgett, and Sanjukta Bhanja. MRAM PUF: A Novel Geometry Based Magnetic PUF With Integrated CMOS. *IEEE Transactions on Nanotechnology*, 14(3):436–443, 2015.

[32] Joep A. de Groot, Boris Skoric, Niels de Vreede, and Jean-Paul M. G. Linnartz. Quantization in Zero Leakage Helper Data Schemes. *EURASIP Journal on Advances in Signal Processing*, 2016:54, 2016.

[33] Jeroen Delvaux. *Security Analysis of PUF-based Key Generation and Entity Authentication*. PhD thesis, Katholieke Universiteit Leuven, Belgium, 2017.

[34] Jeroen Delvaux, Dawu Gu, Dries Schellekens, and Ingrid Verbauwhede. Helper Data Algorithms for PUF-Based Key Generation: Overview and Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(6):889–902, June 2015.

[35] Jeroen Delvaux, Dawu Gu, Ingrid Verbauwhede, Matthias Hiller, and Meng-Day Yu. *Efficient Fuzzy Extraction of PUF-Induced Secrets: Theory and Applications*, pages 412–431. Springer Berlin Heidelberg, 2016.

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

[37] Jeroen Delvaux and Ingrid Verbauwhede. Attacking PUF-Based Pattern Matching Key Generators via Helper Data Manipulation. In Josh Benaloh, editor, *Topics in Cryptology – CT-RSA 2014*, pages 106–131, Cham, 2014. Springer International Publishing.

[38] Jeroen Delvaux and Ingrid Verbauwhede. Key-Recovery Attacks on Various RO PUF Constructions via Helper Data Manipulation. In *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6, 2014.

[39] Claude Desset, Benoît Macq, and Luc Vandendorpe. Block Error-Correcting codes for Systems with a Very High BER: Theoretical Analysis and Application to the Protection of Watermarks. *Signal Processing: Image Communication*, 17(5):409–421, 2002.

[40] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In Christian Cachin and Jan L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 523–540, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[41] Robert F. H. Fischer and Sven Müelich. A New Helper Data Scheme for Soft-Decision Decoding of Binary Physical Unclonable Functions. *IEEE Access*, 10:12644–12653, 2022.

[42] Viktor Fischer, Florent Bernard, Nathalie Bochard, Quentin Dallison, and Maciej Skórski. Enhancing Quality and Security of the PLL-TRNG. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(4):211–237, Aug. 2023.

[43] George David Forney. *Concatenated Codes*. PhD thesis, Massachusetts Institute of Technology. Dept. of Electrical Engineering, December 1965.

[44] **Christoph Frisch and Michael Pehl. Beware of the Bias-Statistical Performance Evaluation of Higher-Order Alphabet PUFs. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1005–1010. IEEE, 2022.**

[45] **Christoph Frisch, Michael Tempelmeier, and Michael Pehl. PAG-IoT: A PUF and AEAD Enabled Trusted Hardware Gateway for IoT Devices. In *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 500–505. IEEE, 2020.**

[46] **Christoph Frisch, Florian Wilde, Thomas Holzner, and Michael Pehl. A Practical Approach to Estimate the Min-Entropy in PUFs. *Journal of Hardware and Systems Security*, Nov 2023.**

[47] Fatemeh Ganji, Domenic Forte, and Jean-Pierre Seifert. PUFmeter a Property Testing Tool for Assessing the Robustness of Physically Unclonable Functions to Machine Learning Attacks. *IEEE Access*, 7:122513–122521, 2019.

[48] Fatemeh Ganji, Shahin Tajik, and Jean-Pierre Seifert. Why Attackers Win: On the Learnability of XOR Arbiter PUFs. In Mauro Conti, Matthias Schunter, and Ioannis Askoxylakis, editors, *Trust and Trustworthy Computing*, pages 22–39, Cham, 2015. Springer International Publishing.

[49] Fatemeh Ganji, Shahin Tajik, and Jean-Pierre Seifert. PAC Learning of Arbiter PUFs. *Journal of Cryptographic Engineering*, 6(3):249–258, 2016.

[50] Kathrin Garb, Marvin Xhemrishi, Ludwig Kürzinger, and Christoph Frisch. The Wiretap Channel for Capacitive PUF-Based Security Enclosures. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(3):165–191, 2022.

[51] Blaise Gassend. Physical Random Functions. Master's thesis, Massachusetts Institute of Technology, January 2003.

[52] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Silicon Physical Random Functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, page 148–160, New York, NY, USA, 2002. Association for Computing Machinery.

[53] Jorge Guajardo, Sandeep S. Kumar, Geert-Jan Schrijen, and Pim Tuyls. FPGA Intrinsic PUFs and Their Use for IP Protection. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, pages 63–80, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[54] Onur Günlü, Onurcan İşcan, Vladimir Sidorenko, and Gerhard Kramer. Code Constructions for Physical Unclonable Functions and Biometric Secrecy Systems. *IEEE Transactions on Information Forensics and Security*, 14(11):2848–2858, 2019.

[55] Kotaro Hayashi, Ryuichi Minagawa, and Naoya Torii. Side-Channel Attack on COSO-Based TRNG to Estimate Output Bits. In *2022 Tenth International Symposium on Computing and Networking Workshops (CANDARW)*, pages 302–308, 2022.

[56] Clemens Helfmeier, Dmitry Nedospasov, Christian Boit, and Seifert Jean-Pierre. Cloning Physically Unclonable Functions. In *Proceedings of the IEEE International Symposium of Hardware-Oriented Security and Trust*. IEEE, June 2013.

[57] **Andreas Herrmann, Michael Weiner, Michael Pehl, and Helmut Graeb. Bringing Analog Design Tools to Security: Modeling and Optimization of a Low Area Probing Detector. In *2018 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pages 1–4. IEEE, 2018.**

[58] Matthias Hiller. *Key Derivation with Physical Unclonable Functions*. Dissertation, Technische Universität München, München, 2016.

[59] Matthias Hiller, Ludwig Kürzinger, and Georg Sigl. Review of Error Correction for PUFs and Evaluation on State-of-the-Art FPGAs. *Journal of Cryptographic Engineering*, 10(3):229–247, Sep 2020.

[60] Matthias Hiller, Dominik Merli, Frederic Stumpf, and Georg Sigl. Complementary IBS: Application Specific Error Correction for PUFs. In *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*, pages 1–6, 2012.

[61] Matthias Hiller, Michael Weiner, Leandro Rodrigues Lima, Maximilian Birkner, and Georg Sigl. Breaking through Fixed PUF Block Limitations with Differential Sequence Coding and Convolutional Codes. In *TrustED*, pages 43–54, 2013.

[62] Matthias Hiller, Meng-Day Yu, and Georg Sigl. Cherry-Picking Reliable PUF Bits With Differential Sequence Coding. *IEEE Transactions on Information Forensics and Security*, 11(9):2065–2076, 2016.

[63] **Matthias Hiller, Meng-Day (Mandel) Yu, and Michael Pehl. Systematic Low Leakage Coding for Physical Unclonable Functions. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pages 155–166, 2015.**

[64] Daniel E. Holcomb, Wayne Burleson, and Kevin Fu. Initial SRAM State as a Fingerprint and Source of True Random Numbers for RFID Tags. In *Proceedings of the Conference on RFID Security*, 2007.

[65] Daniel E. Holcomb, Wayne P. Burleson, and Kevin Fu. Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. *IEEE Transactions on Computers*, 58(9):1198–1210, 2009.

[66] W.T. Holman, J.A. Connelly, and A.B. Dowlatabadi. An Integrated Analog/Digital Random Noise Source. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 44(6):521–528, 1997.

[67] Yohei Hori, Takahiro Yoshida, Toshihiro Katashita, and Akashi Satoh. Quantitative and Statistical Performance Evaluation of Arbiter Physical Unclonable Functions on FPGAs. In *Proceedings of the 2010 International Conference on Reconfigurable Computing and FPGAs*, RECONFIG '10, pages 298–303, Washington, DC, USA, 2010. IEEE Computer Society.

[68] Yiming Huai. Spin-Transfer Torque MRAM (STT-MRAM): Challenges and Prospects. *AAPPS Bulletin*, 18, 01 2008.

[69] Tanya Ignatenko, Geert Jan Schrijen, Boris Skoric, Pim Tuyls, and Frans M. J. Willems. Estimating the Secrecy-Rate of Physical Unclonable Functions with the Context-Tree Weighting Method. In *IEEE International Symposium on Information Theory (ISIT)*, pages 499–503, 2006.

[70] Tanya Ignatenko and Frans Willems. On Privacy in Secure Biometric Authentication Systems. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, volume 2, pages II–121–II–124, 2007.

[71] Vincent Immler, Johannes Obermaier, Martin König, Matthias Hiller, and Georg Sig. B-TREPID: Batteryless Tamper-Resistant Envelope with a PUF and Integrity Detection. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 49–56, April 2018.

[72] Vincent Immler, Johannes Obermaier, Kuan Kuan Ng, Fei Xiang Ke, JinYu Lee, Yak Peng Lim, Wei Koon Oh, Keng Hoong Wee, and Georg Sigl. Secure Physical Enclosures from Covers with Tamper-Resistance. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 51–96, 2019.

[73] Vincent Immler and Karthik Uppund. New Insights to Key Derivation for Tamper-Evident Physical Unclonable Functions. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 30–65, 2019.

[74] Vincent Charles Immler. *Higher-Order Alphabet Physical Unclonable Functions*. PhD thesis, Technische Universität München, 2019.

[75] Intrinsic ID. White Paper: SRAM PUF: The Secure Silicon Fingerprint, 2023.

[76] **Nisha Jacob, Jakob Wittmann, Johann Heyszl, Robert Hesselbarth, Florian Wilde, Michael Pehl, Georg Sigl, and Kai Fischer. Securing FPGA SoC Configurations Independent of Their Manufacturers. In *30th IEEE International System-on-Chip Conference, SOCC 2017, Munich, Germany, September 5-8, 2017*, pages 114–119, 2017.**

[77] Hao Jiang, Daniel Belkin, Sergey E. Savel'ev, Siyan Lin, Zhongrui Wang, Yunning Li, Saumil Joshi, Rivu Midya, Can Li, Mingyi Rao, Mark Barnell, Qing Wu, J. Joshua Yang, and Qiangfei Xia. A Novel True Random Number Generator Based on a Stochastic Diffusive Memristor. *Nature Communications*, 8(1):882, Oct 2017.

[78] Ari Juels and Martin Wattenberg. A Fuzzy Commitment Scheme. In *ACM Conference on Computer and Communications Security (CCS)*, pages 28–36. ACM, 1999.

[79] Deniz Karakoyunlu and Berk Sunar. Differential Template Attacks on PUF Enabled Cryptographic Devices. In *2010 IEEE International Workshop on Information Forensics and Security*, pages 1–6, 2010.

[80] **Claus Kestel, Christoph Frisch, Michael Pehl, and Norbert Wehn. Towards More Secure PUF Applications: A Low-Area Polar Decoder Implementation. In *2022 IEEE 35th International System-on-Chip Conference (SOCC)*, pages 1–6, 2022.**

[81] Wolfgang Killmann and Werner Schindler. A Proposal for: Functionality Classes for Random Number Generators Version 2.0, 2011.

[82] Wolfgang Killmann and Werner Schindler. A Proposal for: Functionality Classes for Random Number Generators Version 2.35 – Draft, 2022.

[83] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[84] Paul Kohlbrenner and Kris Gaj. An Embedded True Random Number Generator for FPGAs. In *Proceedings of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays*, FPGA '04, page 71–78, New York, NY, USA, 2004. Association for Computing Machinery.

[85] Oliver Kömmerling and Markus G. Kuhn. Design Principles for Tamper-resistant Smartcard Processors. In *Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology*, WOST'99, pages 2–2, Berkeley, CA, USA, 1999. USENIX Association.

[86] Trevor Kroeger, Wei Cheng, Sylvain Guilley, Jean-Luc Danger, and Naghmeh Karimi. Cross-PUF Attacks on Arbiter-PUFs through their Power Side-Channel. In *2020 IEEE International Test Conference (ITC)*, pages 1–5, 2020.

[87] Fei Li and Jayce Lay Keng Lim. ReRAM Non-Volatile AES Encryption Engine for IoT Application. In *2019 32nd IEEE International System-on-Chip Conference (SOCC)*, pages 359–364, 2019.

[88] Daihyun Lim. Extracting Secret Keys from Integrated Circuits. Massachusetts Institute of Technology, May 2004. Master Thesis.

[89] **Bernhard Lippmann, Joel Hatsch, Stefan Seidl, Detlef Houdeau, Niranjana Papagudi Subrahmanyam, Daniel Schneider, Malek Safieh, Anne Passarelli, Aliza Maftun, Michaela Brunner, et al. VE-FIDES: Designing Trustworthy Supply Chains Using Innovative Fingerprinting Implementations. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2023.**

[90] Y. A. Liu, L. Chen, X. W. Li, Y. L. Liu, S. G. Hu, Q. Yu, T. P. Chen, and Y. Liu. A Dynamic AES Cryptosystem Based on Memristive Neural Network. *Scientific Reports*, 12(1):12983, Jul 2022.

[91] K. Lofstrom, W.R. Daasch, and D. Taylor. IC Identification Circuit Using Device Mismatch. In *2000 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No.00CH37056)*, pages 372–373, 2000.

[92] Heiko Lohrke, Shahin Tajik, Christian Boit, and Jean-Pierre Seifert. No Place to Hide: Contactless Probing of Secret Data on FPGAs. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems – CHES 2016*, pages 147–167, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[93] Heiko Lohrke, Shahin Tajik, Thilo Krachenfels, Christian Boit, and Jean-Pierre Seifert. Key Extraction Using Thermal Laser Stimulation: A Case Study on Xilinx Ultrascale FPGAs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 573–595, 2018.

[94] Roel Maes. *Physically Unclonable Functions: Constructions, Properties and Applications*. Dissertation, Katholieke Universiteit Leuven, 2012.

[95] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. Low-Overhead Implementation of a Soft Decision Helper Data Algorithm for SRAM PUFs. In Christophe Clavier and Kris Gaj, editors, *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pages 332–347. Springer, Heidelberg, 2009.

[96] Roel Maes, Anthony Van Herrewege, and Ingrid Verbauwhede. PUFKY: A Fully Functional PUF-Based Cryptographic Key Generator. In Emmanuel Prouff and Patrick Schaumont, editors, *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 7428 of *LNCS*, pages 302–319. Springer, Heidelberg, 2012.

[97] Abhranil Maiti, Jeff Casarona, Luke McHale, and Patrick Schaumont. A Large Scale Characterization of RO-PUF. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 66–71, 2010.

[98] Abhranil Maiti, Vikash Gunreddy, and Patrick Schaumont. A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions. Cryptology ePrint Archive, Report 2011/657, 2011. http://eprint.iacr.org/2011/657.

[99] Mehrdad Majzoobi, Masoud Rostami, Farinaz Koushanfar, Dan S. Wallach, and Srinivas Devadas. Slender PUF Protocol: A Lightweight, Robust, and Secure Authentication by Substring Matching. In *International Workshop on Trustworthy Embedded Devices (TrustED)*, pages 33–44, 2012.

[100] Salvador Manich and Martin Strasser. A Highly Time Sensitive XOR Gate for Probe Attempt Detectors. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 60-II(11):786–790, 2013.

[101] A. Theodore Markettos and Simon W. Moore. The Frequency Injection Attack on Ring-Oscillator-Based True Random Number Generators. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 317–331. Springer, 2009.

[102] Jonathan Masci, Michael M. Bronstein, Alexander M. Bronstein, and Jürgen Schmidhuber. Multimodal Similarity-Preserving Hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):824–830, 2014.

[103] Sanu K. Mathew, Sudhir K. Satpathy, Patrick Koeberl, Jiangtao Li, Ram K. Krishnamurthy, and Anand Rajan. Using Dark Bits to Reduce Physical Unclonable Function (PUF) Error Rate Without Storing Dark Bits Location, 2013.

[104] Thomas McGrath, Ibrahim E. Bagci, Zhiming M. Wang, Utz Roedig, and Robert J. Young. A PUF Taxonomy. *Applied Physics Reviews*, 6(1):011303, 2019.

[105] Nele Mentens, Jan Genoe, Thomas Vandenabeele, Lynn Verschueren, Dirk Smets, Wim Dehaene, and Kris Myny. Security on Plastics: Fake or Real? *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(4):1–16, Aug. 2019.

[106] Dominik Merli, Johann Heyszl, Benedikt Heinz, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Localized Electromagnetic Analysis of RO PUFs. In *Proceedings of the IEEE Int. Symposium of Hardware-Oriented Security and Trust*. IEEE, June 2013.

[107] Dominik Merli, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Semi-Invasive EM Attack on FPGA RO PUFs and Countermeasures. In *6th Workshop on Embedded Systems Security (WESS'2011)*, Taipei, Taiwan, October 2011. ACM.

[108] Dominik Merli, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Side-Channel Analysis of PUFs and Fuzzy Extractors. In Jonathan M. McCune, Boris Balacheff, Adrian Perrig, Ahmad-Reza Sadeghi, Angela Sasse, and Yolanta Beres, editors, *International Conference on Trust and Trustworthy Computing (TRUST)*, volume 6740 of *LNCS*, pages 33–47. Springer, 2011.

[109] Dominik Merli, Frederic Stumpf, and Georg Sigl. Protecting PUF Error Correction by Codeword Masking. Cryptology ePrint Archive, Paper 2013/334, 2013. `https://eprint.iacr.org/2013/334`.

[110] **Seyed Hamidreza Moghadas and Michael Pehl. ROPAD: A Fully Digital Highly Predictive Ring Oscillator Probing Attempt Detector. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.**

[111] **Seyed Hamidreza Moghadas, Michael Pehl, and Georg Sigl. ROPAD +: Enhancing the Digital Ring Oscillator Probing Attempt Detector for Protecting Irregular Data Buses. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2022.**

[112] Ugo Mureddu, Nathalie Bochard, Lilian Bossuet, and Viktor Fischer. Experimental Study of Locking Phenomena on Oscillating Rings Implemented in Logic Devices. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(7):2560–2571, 2019.

[113] Ugo Mureddu, Brice Colombier, Nathalie Bochard, Lilian Bossuet, and Viktor Fischer. Transient Effect Ring Oscillators Leak Too. In *2019 IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2019, Miami, FL, USA, July 15-17, 2019*, pages 37–42. IEEE, 2019.

[114] Kalle Ngo and Elena Dubrova. Side-Channel Analysis of the Random Number Generator in STM32 MCUs. In *Proceedings of the Great Lakes Symposium on VLSI 2022*, GLSVLSI '22, page 15–20, New York, NY, USA, 2022. Association for Computing Machinery.

[115] Phuong Ha Nguyen, Durga Prasad Sahoo, Chenglu Jin, Kaleel Mahmood, Ulrich Rührmair, and Marten van Dijk. The Interpose PUF: Secure PUF Design against State-of-the-art Machine Learning Attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(4):243–290, Aug. 2019.

[116] Saki Osuka, Daisuke Fujimoto, Shinichi Kawamura, and Yuichi Hayashi. Electromagnetic Side-Channel Analysis Against TERO-Based TRNG. *IEEE Transactions on Electromagnetic Compatibility*, 64(5):1288–1295, October 2022.

[117] Ravikanth Pappu. *Physical One-Way Functions*. PhD thesis, Massachusetts Institute of Technology, 2001.

[118] Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical One-Way Functions. *Science*, 297(5589):2026–2030, September 2002.

[119] Adriaan Peetermans, Vladimir Rozic, and Ingrid Verbauwhede. A Highly-Portable True Random Number Generator Based on Coherent Sampling. In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, pages 218–224, 2019.

[120] **Michael Pehl, Christoph Frisch, Peter Christian Feist, and Georg Sigl. KeLiPUF: A Key-Distribution Protocol for Lightweight Devices Using Physical Unclonable Functions. In *17^{th} escar Europe : embedded security in cars*. 2019.**

[121] **Michael Pehl, Matthias Hiller, and Helmut Gräb. Efficient Evaluation of Physical Unclonable Functions Using Entropy Measures. In *Journal of Circuits, Systems and Computers*, volume 25, pages 1640001:1–1640001:23, 2016.**

[122] **Michael Pehl, Matthias Hiller, and Georg Sigl. *Information Theoretic Security and Privacy of Information Systems*, chapter Secret Key Generation for Physical Unclonable Functions, pages 362–389. Cambridge University Press, 2017.**

[123] **Michael Pehl, Akshara Ranjit Punnakkal, Matthias Hiller, and Helmut Graeb. Advanced Performance Metrics for Physical Unclonable Functions. In *International Symposium on Integrated Circuits (ISIC)*. IEEE, 2014.**

[124] **Michael Pehl, Tobias Tretschok, Daniel Becker, and Vincent Immler. Spatial Context Tree Weighting for Physical Unclonable Functions. In *2020 European Conference on Circuit Theory and Design (ECCTD)*, pages 1–4. IEEE, 2020.**

[125] **Michael Pehl, Florian Wilde, Berndt M. Gammel, and Georg Sigl. Qualitätsevaluierung von Physical Unclonable Functions als Schlüsselspeicher. In *14. Deutscher IT-Sicherheitskongress*, 2015.**

[126] Vikash Kumar Rai, Somanath Tripathy, and Jimson Mathew. Memristor Based Random Number Generator: Architectures and Evaluation. *Procedia Computer Science*, 125:576–583, 2018. The 6th International Conference on Smart Computing and Communications.

[127] Jeyavijayan Rajendran, Garrett S. Rose, Ramesh Karri, and Miodrag Potkonjak. Nano-PPUF: A Memristor-Based Security Primitive. In *2012 IEEE Computer Society Annual Symposium on VLSI*, pages 84–87, Aug 2012.

[128] **Carl Riehm, Christoph Frisch, Florin Burcea, Matthias Hiller, Michael Pehl, and Ralf Brederlow. Structured Design and Evaluation of a Resistor-Based PUF Robust Against PVT-Variations. In *2023 26th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, pages 93–98. IEEE, 2023.**

[129] Garrett S. Rose. Overview: Memristive Devices, Circuits and Systems. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 1955–1958, 2010.

[130] Garrett S. Rose, Nathan McDonald, Lok-Kwong Yan, and Bryant Wysocki. A Write-Time Based Memristive PUF for Hardware Security Applications. In *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 830–833, Nov 2013.

[131] **Jonas Ruchti, Michael Gruber, and Michael Pehl. When the Decoder Has to Look Twice: Glitching a PUF Error Correction. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 26–70, 2022.**

[132] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. Modeling Attacks on Physical Unclonable Functions. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 237–249, New York, NY, USA, 2010. ACM.

[133] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, N. Heckert, James Dray, San Vo, and Lawrence Bassham. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. In *Special Publication (NIST SP) - 800-22 Rev. 1*. National Institute of Standards and Technology, 2010.

[134] Durga Prasad Sahoo, Debdeep Mukhopadhyay, Rajat Subhra Chakraborty, and Phuong Ha Nguyen. A Multiplexer-Based Arbiter PUF Composition with Enhanced Reliability and Security. *IEEE Transactions on Computers*, 67(3):403–417, 2018.

[135] Pranesh Santikellur, Aritra Bhattacharyay, and Rajat Subhra Chakraborty. Deep Learning Based Model Building Attacks on Arbiter PUF Compositions. *Cryptology ePrint Archive, Paper*, page 566, 2019.

[136] Sudhir Satpathy, Sanu Mathew, Jiangtao Li, Patrick Koeberl, Mark Anders, Himanshu Kaul, Gregory Chen, Amit Agarwal, Steven Hsu, and Ram Krishnamurthy. 13fJ/Bit Probing-Resilient 250K PUF Array with Soft Darkbit Masking for 1.94% Bit-Error in 22nm Tri-Gate

CMOS. In *ESSCIRC 2014 - 40th European Solid State Circuits Conference (ESSCIRC)*, pages 239–242, 2014.

[137] Kaveh Shamsi and Yier Jin. Security of Emerging Non-Volatile Memories: Attacks and Defenses. In *2016 IEEE 34th VLSI Test Symposium (VTS)*, pages 1–4, 2016.

[138] Mitsuru Shiozaki and Takeshi 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*, ASHES'19, page 13–21, New York, NY, USA, 2019. Association for Computing Machinery.

[139] Mitsuru Shiozaki, Yohei Hori, and Takeshi Fujino. Entropy Estimation of Physically Unclonable Functions with Offset Error. Cryptology ePrint Archive, Paper 2020/1284, 2020. `https://eprint.iacr.org/2020/1284`.

[140] **Emanuele Strieder, Christoph Frisch, and Michael 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*, pages 1–36, 2021.**

[141] Dmitri B Strukov, Gregory S Snider, Duncan R Stewart, and R Stanley Williams. The Missing Memristor Found. *Nature*, 453(7191):80–83, May 2008.

[142] Ying Su, Jeremy Holleman, and Brian P. Otis. A Digital 1.6 pJ/bit Chip Identification Circuit Using Process Variations. *IEEE Journal OF Solid-State Circuits*, 43(1):69–77, Jan 2008.

[143] Gookwon Edward Suh and Srinivas Devadas. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *ACM/IEEE Design Automation Conference (DAC)*, pages 9–14, 2007.

[144] Sven Müelich and Sven Puchinger and Martin Bossert. Constructing an ldpc code containing a given vector, 2018.

[145] Shahin Tajik, Enrico Dietz, Sven Frohmann, Helmar Dittrich, Dmitry Nedospasov, Clemens Helfmeier, Jean-Pierre Seifert, Christian Boit, and Heinz-Wilhelm Hübers. Photonic Side-Channel Analysis of Arbiter PUFs. *Journal of Cryptology*, 30(2):550–571, 2017.

[146] Shahin Tajik, Enrico Dietz, Sven Frohmann, Jean-Pierre Seifert, Dmitry Nedospasov, Clemens Helfmeier, Christian Boit, and Helmar Dittrich. Physical Characterization of Arbiter PUFs. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, pages 493–509, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[147] Shahin Tajik, Heiko Lohrke, Fatemeh Ganji, Jean-Pierre Seifert, and Christian Boit. Laser Fault Attack on Physically Unclonable Functions. In *2015 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 85–96, 2015.

[148] Lars Tebelmann. *Side-Channel Analysis and Countermeasures for Physical Unclonable Functions*. Dissertation, Technische Universität München, München, 2023.

[149] **Lars Tebelmann, Jean-Luc Danger, and Michael Pehl. Self-Secured PUF: Protecting the Loop PUF by Masking. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 293–314. Springer, 2020.**

[150] **Lars Tebelmann, Jean-Luc Danger, and Michael Pehl. Interleaved Challenge Loop PUF: A Highly Side-Channel Protected Oscillator-Based PUF. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 69(12):5121–5134, 2022.**

[151] **Lars Tebelmann, Ulrich Kühne, Jean-Luc Danger, and Michael Pehl. Analysis and Protection of the Two-Metric Helper Data Scheme. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 279–302. Springer, 2021.**

[152] **Lars Tebelmann, Michael Pehl, and Vincent Immler. Side-Channel Analysis of the TERO PUF. In Ilia Polian and Marc Stöttinger, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 43–60, Cham, 2019. Springer International Publishing.**

[153] **Lars Tebelmann, Michael Pehl, and Georg 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*, ASHES '17, pages 43–52, New York, NY, USA, 2017. ACM.**

[154] **Lars Tebelmann, Moritz Wettermann, and Michael Pehl. On-Chip Side-Channel Analysis of the Loop PUF. In *Proceedings of the 2022 Workshop on Attacks and Solutions in Hardware Security*, ASHES'22, pages 55 – 63, New York, NY, USA, 2022. Association for Computing Machinery.**

[155] Meltem Sonmez Turan, Elaine B. Barker, John M. Kelsey, Kerry A. McKay, Mary L. Baish, and Mike Boyle. Recommendation for the Entropy Sources Used for Random Bit Generation. In *Special Publication (NIST SP) - 800-90B*. National Institute of Standards and Technology, 2018.

[156] Pim Tuyls and Boris Škorić. *Strong Authentication with Physical Unclonable Functions*, pages 133–148. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[157] Vincent Van der Leest, Bart Preneel, and Erik Van der Sluis. Soft Decision Error Correction for Compact Memory-Based PUFs Using a Single Enrollment. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 268–282. Springer, 2012.

[158] Michal Varchola and Milos Drutarovsky. New High Entropy Element for FPGA Based True Random Number Generators. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, pages 351–365, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[159] Sascha Vongehr and Xiangkang Meng. The Missing Memristor has Not been Found. *Scientific Reports*, 5(1):11657, Jun 2015.

[160] Michael Weiner, Salvador Manich, Rosa Rodríguez-Montañés, and Georg Sigl. The Low Area Probing Detector as a Countermeasure Against Invasive Attacks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(2):392–403, 2017.

[161] Michael Weiner, Wolfgang Wieser, Emili Lupon, Georg Sigl, and Salvador Manich. A Calibratable Detector for Invasive Attacks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(5):1067–1079, 2019.

[162] Florian Wilde. Large Scale Characterization of SRAM on Infineon XMC Microcontrollers As PUF. In *Proceedings of the Fourth Workshop on Cryptography and Security in Computing Systems*, CS2 '17, pages 13–18, New York, NY, USA, 2017. ACM.

[163] Florian Wilde. *Metrics for Physical Unclonable Functions*. PhD thesis, Technische Universität München, 2021.

[164] **Florian Wilde, Christoph Frisch, and Michael Pehl. Efficient Bound for Conditional Min-Entropy of Physical Unclonable Functions Beyond IID. In *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2019.**

[165] **Florian Wilde, Berndt M. Gammel, and Michael Pehl. Spatial Correlation Analysis on Physical Unclonable Functions. *IEEE Transactions on Information Forensics and Security*, 13(6):1468–1480, June 2018.**

[166] **Florian Wilde, Matthias Hiller, and Michael Pehl. Statistic-Based Security Analysis of Ring Oscillator PUFs. In *2014 International Symposium on Integrated Circuits (ISIC)*, pages 148–151, December 2014.**

[167] **Florian Wilde and Michael Pehl. On the Confidence in Bit-Alias Measurement of Physical Unclonable Functions. In *2019 17th IEEE International New Circuits and Systems Conference (NEWCAS)*, pages 1–4. IEEE, 2019.**

[168] Frans M. J. Willems, Yu M. Shtarkov, and Tjalling J. Tjalkens. Context Tree Weighting : A Sequential Universal Source Coding Procedure for FSMX Sources. In *Proceedings. IEEE International Symposium on Information Theory*, pages 59–59, Jan 1993.

[169] Frans M. J. Willems, Yu M. Shtarkov, and Tjalling J. Tjalkens. The Context-Tree Weighting Method: Basic Properties. *Transactions in Information Theory*, pages 653–664, 1995.

[170] Nils Wisiol, Christopher Mühl, Niklas Pirnay, Phuong Ha Nguyen, Marian Margraf, Jean-Pierre Seifert, Marten van Dijk, and Ulrich Rührmair. Splitting the Interpose PUF: A Novel Modeling Attack Strategy. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):97–120, 2020.

[171] Kon Max Wong and Shuang Chen. The Entropy of Ordered Sequences and Order Statistics. *IEEE Transactions on Information Theory*, 36(2):276–284, 1990.

[172] Meng-Day Yu and Srinivas Devadas. Secure and Robust Error Correction for Physical Unclonable Functions. *IEEE Design & Test of Computers*, 27(1):48–65, 2010.

[173] Meng-Day Yu, Matthias Hiller, Jeroen Delvaux, Richard Sowell, Srinivas Devadas, and Ingrid Verbauwhede. A Lockdown Technique to Prevent Machine Learning on PUFs for Lightweight Authentication. *IEEE Transactions on Multi-Scale Computing Systems*, 2(3):146–159, July 2016.

[174] Meng-Day Yu, David M'Raïhi, Ingrid Verbauwhede, and Srinivas Devadas. A Noise Bifurcation Architecture for Linear Additive Physical Functions. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 124–129, May 2014.