Chapter 4
Implementation and Experimental Analysis of Intrinsic PUFs

4.1 Introduction

4.1.1 Motivation

All currently known intrinsic PUF implementations are silicon-based and obtain their PUF behavior from process variations during the manufacturing of silicon chips. These PUFs are of particular interest, because their response values can be used as a secret element in a larger security implementation on the same silicon chip. Deploying an intrinsic PUF in a security application in this way provides interesting practical and security advantages. We will discuss applications of intrinsic PUFs and their added value in great detail, respectively for PUF-based authentication in Chap. 5 and PUF-based key generation in Chap. 6.

The realization that a silicon intrinsic PUF can serve as an integral hardware security primitive with valuable properties such as uniqueness and unpredictability, and in particular physical unclonability, which cannot be obtained solely from algorithmic constructions, has led to a great interest into its constructions. Many intrinsic PUF implementations were proposed over time and are discussed in detail in Sect. 2.4. When one actually wants to deploy an intrinsic PUF in a hardware security system, interest goes out to the levels of efficiency and performance of all these different constructions, both from a PUF perspective (which construction shows the best PUF behavior) as well as from a typical hardware design perspective (which construction offers the lowest area, highest speed, lowest power use, etc.). Section 2.4.8 provides an overview of known experimental results concerning PUF behavior, but it is also made clear that a comparison of different proposals solely based on these results is not entirely objective, for a number of reasons:

1. For all known intrinsic PUF constructions, as for nearly all hardware security primitives, there exists a trade-off between area/speed and security, i.e. larger and/or slower implementations typically offer higher levels of security. In the overview in Sect. 2.4.8, which lists experimental PUF results available in literature, some of the considered implementations are heavily biased towards opti-

mizing their PUF behavior by greatly sacrificing on implementation area and/or speed, while others are not.
2. Experimental PUF results always focus on two properties, reliability and uniqueness, which are mostly summarized by calculating the average response intra-distance ($\mu_{\mathcal{P}}^{\text{intra}}$) and inter-distance ($\mu_{\mathcal{P}}^{\text{inter}}$) of the experiment. Good PUF behavior is expressed by a small $\mu_{\mathcal{P}}^{\text{intra}}$ and a large $\mu_{\mathcal{P}}^{\text{inter}}$. However, it is not immediately clear how to combine both measures into a single quality parameter for a particular PUF construction.
3. The results presented in Sect. 2.4.8 come from implementations using a variety of different technologies and platforms. It is typically hard to accurately scale implementation results such as area and speed from one technology to another, e.g. between different CMOS technology nodes, and scaling between different platforms, e.g. from FPGA to ASIC, can only be done very roughly. Extrapolation of PUF behavior results to different technologies and platforms is virtually impossible.

The first two issues cannot be dealt with for bare PUFs, but need to be considered in the application context of the bigger system deploying the PUF. The optimal trade-off between security and efficiency, as well as the relation between reliability and uniqueness, are determined by the requirements and constraints of the security system as a whole. Optimizing the PUF implementation is an entangled part of a bigger design optimization process which will be discussed in detail for PUF-based authentication systems in Chap. 5 and for PUF-based cryptographic key generation in Chap. 6.

The last issue can be approached by implementing different intrinsic PUF proposals on the same platform and using the same technology. This will be the main topic of this chapter. A selection is made of intrinsic PUF proposals which were proven to show acceptable PUF behavior, and a number of instantiations of each of them is integrated in an ASIC design. This ASIC design is processed and a significant set of silicon chips implementing it is manufactured. Based on this set of devices, valuable experimental PUF data is gathered which can be cross-compared without restraint, since it results from PUF constructions implemented on the same silicon die.

Background Designing and manufacturing an ASIC is a complex, time-consuming, and costly undertaking with significant risk of failure. The findings presented in this chapter are the joint successful result of a European research project called ‘UNIQUE’ in which the author’s institution was a partner [149]. We definitely want to acknowledge the other project partners which contributed heavily to the design, production and evaluation of the test chip discussed in this chapter.

4.1.2 Chapter Goals

The main goal of this chapter is to produce a practical and an objective analysis and comparison of different intrinsic PUF constructions, by implementing them on the same platform, evaluating them under the same conditions, and assessing them on the same characteristics. In this chapter we plan to:

- Discuss the design process and implementation details of a custom test chip carrying instantiations of six different intrinsic PUFs.
- Present an in-depth analysis of the uniqueness and reproducibility of the evaluation results of the test chip implementations, including the influence of varying evaluation conditions. This analysis should result in a practically usable characterization of these properties for every PUF instance, which can be immediately plugged into the design and optimization process of an application seeking to deploy one of these PUF implementations.
- Discuss the notion of PUF response entropy and introduce a number of practical entropy bounds which can be computed based on experimentally obtained response evaluations, including a method to calculate a PUF response entropy bound based on the results of a modeling attack.

4.1.3 Chapter Overview

Section 4.2 discusses the realization of the intrinsic PUF test chip, from its initial design rationale and requirements, through its architecture and the description of its building blocks, to its manufacturing flow details. In Sect. 4.3, we describe how the test chip samples were evaluated and we present a complete analysis of uniqueness and reproducibility based on a large data set of evaluation results. The entropy of a PUF response is described in Sect. 4.4 as a measure of its unpredictability. We introduce a number of increasingly tighter bounds on the response entropy of a PUF, based on considering increasingly more advanced adversary models. Finally, the main results of this chapter are summarized in Sect. 4.5.

4.2 Test Chip Design

4.2.1 Design Rationale

The rationale behind the design of the test chip is guided by two main considerations:

(i) In the end, the goal of the test chip is to collect statistically significant experimental data from intrinsic PUF implementations. Ideally, we would like to implement as many and as large instances as possible from as many different intrinsic PUF proposals as possible and evaluate them in a quick, easy and realistic manner, taking into account that the available area budget should be more or less evenly distributed among the different PUFs.

(ii) Designing and producing an ASIC is a very complex process with a minimal margin for error. The probability of critical failures increases steadily with the size and complexity of the design. Since the coordinating project provides only a single opportunity for ASIC production, it needs to be first-time-right and any risk of failure should be minimized.

These two considerations lead to the following design choices:

- To minimize risk, the overall architecture is kept minimalistic, with the major portion of the silicon area budget devoted to implementations of PUF instances.
- We mainly choose to implement PUF constructions which (at the time) were proven to show PUF behavior in earlier experiments.
- The additional components are kept to the bare minimum required to realistically evaluate the PUFs. This leaves the most area to the actual PUF implementations. All measurement post-processing is done off-line.
- The measurement communication interface, being a single-point-of-failure in the whole design, is kept as simple as possible to minimize all risk. This comes at a significant sacrifice in measurement speed.
- Whenever possible, standard design flows are used. Except for the low-level implementation of some of the PUFs, the whole design is described at the RTL level and synthesized using reliable third-party standard cell libraries for the considered technology.

4.2.2 Design Requirements

PUF Selection

Six intrinsic PUF structures are selected for integration on the test chip:

1. The ring oscillator PUF as proposed by Suh and Devadas [136].
2. The latch PUF as proposed by Su et al. [135].
3. The SRAM PUF as proposed by Guajardo et al. [45].
4. The D Flip-Flop PUF as proposed by Maes et al. [84].
5. The arbiter PUF as proposed by Lee et al. [75].
6. The buskeeper PUF as proposed by Simons et al. [132].

The first five PUF constructions are selected because they were proven to show PUF behavior in earlier implementations. The buskeeper PUF is a newly proposed PUF construction by Simons et al. [132]. For the ring oscillator and the arbiter PUF structures, we analyse two different evaluation methods (cf. Sect. 4.3.1) resulting in a total of eight different PUF constructions on the chip.

Evaluation Control

Additional control over the evaluation conditions of the selected PUF implementations is desirable:

- In order to study the effects of the power-up conditions on the PUF constructions which depend on power-up behavior, a number of PUF instances are grouped under a separate power supply on the chip. This allows us to test instances on the same chip under different supply voltage conditions.
- In a realistic application, a PUF is integrated on the same silicon die implementing the complete hardware system, including a large amount of rapidly switching logic. This might have an effect on the PUF's behavior, e.g. because it introduces switching noise on the supply voltage. To mimic this behavior, we implement an active core on the test chip whose sole purpose is to generate switching activity while the PUFs are evaluated.

Interfacing

To transfer the measurement data off-chip we require a communication interface which offers reasonable transfer rates at minimal design complexity and low pin-count. We prefer a standardized interface for easy integration with other components. A Serial Peripheral Interface (SPI) [98] was selected.

Internally, the different building blocks on the chip need to be accessible in a straightforward manner. However, since all blocks operate in slave-mode, we don't require advanced communication control and we don't want to dedicate silicon area to complex bus interfaces. We opted for a memory-mapped organization, with a single address decoder controlling which building block is being read from or written to. Since most selected PUF blocks are inherently memory elements, they are trivially integrated in this organization. For the other building blocks, input and output ports are being accessed through addressable registers.

4.2.3 Top-Level Architecture

Figure 4.1 indicates the top-level architecture of the test chip design. All data communication is done through the SPI/memory-mapped interface, except for the basic clock and power domain controls which has dedicated control and status pins. The SPI encoder and decoder is a standard design supporting the SPI protocol. The active core is basically an implementation of a large number of unrolled rounds of a random substitution/permutation layer, with the only intention of generating a lot of switching activity. The memory mapper consists of a large multiplexer and demultiplexer which direct data to and from the addressed building block. Internally, data interfaces are 32-bit signals and the address is a 19-bit signal.

Figure 4.2 details the memory map of the different building blocks onto the address space. The three most significant address bits are used to select a particular building block. The next four address bits select a particular instance within the building block, e.g. a particular instance of a PUF type. The remaining 12 address bits are used for addressing within a single instance.

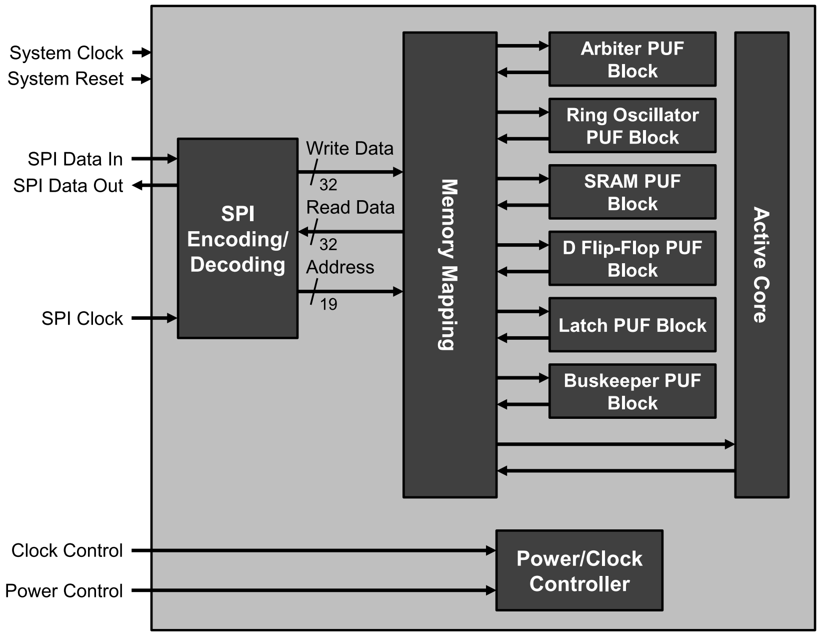
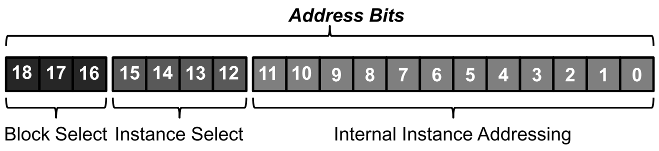


Fig. 4.1 Top-level block diagram of the test chip



Block Select	Block Name
0x0	Ring Oscillator PUF Block
0x1	Latch PUF Block
0x2	SRAM PUF Block
0x3	D Flip-flop PUF Block
0x4	Arbiter PUF Block
0x5	Buskeeper PUF Block
0x6	Active Core

Instance Select	Block Name
0x0 – 0x7	Select PUF Instance 0...7
0xF	Select PUF Controller

Fig. 4.2 Address structure of the internal memory map of the test chip

4.2.4 PUF Block: Arbiter PUF

We design the arbiter PUF according to the original construction from Lee et al. [75], as shown in Fig. 2.1. The switch blocks are constructed using two 2-to-1 multiplexers. The arbiter is an SR latch consisting of two cross-coupled NAND-gates. Each arbiter PUF has a delay chain consisting of 64 concatenated switch blocks, which is long enough to accumulate sufficient delay randomness in order to exhibit an observable difference between the two lines and with high probability avoid the arbiter going into the metastable state. This also means the arbiter PUF takes 64-bit challenges. Using more switch blocks gives even longer challenges, but does not substantially increase the arbiter PUF's unpredictability and does result in larger area use and slower evaluation. To minimize bias in the delay lines and in the arbiter circuit, the whole arbiter PUF is a full custom design, i.e. all design steps including the geometrical sizing, placement and routing of the transistors and interconnecting metal lines are done by hand. Special attention is paid to the symmetry of the arbiter circuit and to balancing the parasitic capacitances of the delay lines as closely as possible. The test chip contains 256 instantiations of this arbiter PUF design, which are grouped into eight instances of 32 arbiter PUFs each. This grouping is only for evaluation performance reasons (32 arbiter PUF response bits can be read out simultaneously over the 32-bit data bus), since all instantiations are identical.

4.2.5 PUF Block: Ring Oscillator PUF

The design of the ring oscillator PUF is based on the construction from Suh and Devadas [136] which is depicted in Fig. 2.3. A ring oscillator consists of 80 chained inverters and one NAND gate to control the oscillation. The number of looped inverters roughly determines the nominal frequency of the ring oscillator, and in the order of 60∼80 inverters are required to obtain frequencies in the range of 500∼700 MHz, which are countable with regular digital counters in the targeted technology. Of this ring oscillator, 4096 identical copies are implemented on the test chip, arranged in 16 batches of 256 oscillators each. Every batch has a single frequency counter and a 256-to-1 multiplexer connects one of the batch's oscillators to the counter. To cope with the high frequency oscillations, the counter is implemented as a 32-bit toggle counter which has a very short critical path. Since there are 16 batches, each with its own counter, 16 oscillation frequencies can be measured in parallel. The measurement time during which oscillations are counted is determined as a particular number of oscillations of an independent, slightly faster, oscillator (64 inverters + NAND) which feeds a timer. The exact number of oscillations of this timing loop can be set by the user. The actual response bit generation is not implemented on the test chip, but the counter values are measured directly. The response bit generation based on the measured frequencies is performed off-line, using an algorithm of one's choice. The response evaluation methods we use are detailed in Sect. 4.3.1.

4.2.6 PUF Block: SRAM PUF

The SRAM PUF, as proposed by Guajardo et al. [45], basically consists of standard SRAM cells of which the power-up value is measured. We implement an SRAM

PUF using a third-party (TSMC) SRAM IP block implementing an addressable array of 2048×32 SRAM cells, each cell consisting of six MOSFETs. Each of these blocks can generate 65536 response bits (64 kbit). Four of these SRAM PUF instances are placed on the test chip.

4.2.7 PUF Blocks: D Flip-Flop PUF, Latch PUF and Buskeeper PUF

The design of these three PUFs basically consists of instantiations of the basic elements: D flip-flops, latches and buskeeper cells. For our test chip, the operation of all three of these PUFs is based on the power-up behavior of their basic cells. The design of each of these cells comes from a third-party (TSMC) standard cell library. The only other difference between these PUFs is the number of cells which are instantiated and the way the cells are organized in arrays.

D Flip-Flop PUF Block

One D flip-flop PUF is designed containing 8192 D flip-flop standard cells. Four of these PUFs are instantiated on the test chip. In the first two instances, the D flip-flops are organized in a long scan chain, allowing them to be read out sequentially. In the last two instances, the flip-flops are organized in a large multiplexer tree, allowing them to be addressed individually. The flip-flops in the multiplexer tree have their data inputs connected to the write input of the memory map, which is grounded when the D flip-flop PUF is not addressed.

Latch PUF

We design a latch PUF which consists of 8192 standard cell latches, and four of these latch PUFs instances are implemented on the test chip. Again, the first two instances have a scan chain-based organization while the latter are organized in a multiplexer tree. For latches, the scan chain design is slightly more advanced. Because latches are level-triggered, as opposed to flip-flops which are edge-triggered, they cannot be all clocked at the same time to shift their values in a chain. The latches in the multiplexer tree have their data inputs grounded.

Buskeeper PUF

The buskeeper PUF design contains 8192 buskeeper cells, and two of the buskeeper PUFs are instantiated on the test chip. Both are organized in an addressable multiplexer tree, since buskeeper cells cannot be chained (they have only a single-bit bidirectional port).

Practical Comparison

D flip-flop, latch and buskeeper PUFs are very similar in design, the main difference being the implementation of their basic cells. The following practical considerations can be made:

- Buskeeper cells, consisting of two inverters, are the smallest of the three, which makes the buskeeper PUF the most area-efficient in terms of response bits per silicon area. However, since they cannot be chained, they do require a (rather large) multiplexer tree to read them out.
- Latches, which consist of two cross-coupled NAND or NOR gates, are larger than buskeeper cells, but smaller than D flip-flops. They can be chained, but this is not trivial.
- D flip-flops are typically constructed from two latches and are therefore the largest of all three basic cells, but they can be easily chained. Moreover, D flip-flops are also a very common cell in regular digital designs, which could make them reusable for other purposes after they have generated their PUF response bit at power-up.

In comparison to SRAM PUFs, these three PUF types are less efficient since SRAM arrays are heavily area-optimized. However, their cell-based design allows us more flexibility since they can be instantiated one cell at a time, while SRAM only comes in bulky arrays. This also allows us to spread, e.g. a D flip-flop PUF, randomly over the whole area of a silicon die, which adds a layer of physical obscurity against optical scrutiny attacks. SRAM arrays on the other hand are easily spotted due to their large and very regular matrix structure.

Besides these practical observations, the PUF behavior of each of these memory-based PUFs should of course also be taken into account. This is the goal of the test chip as discussed in this chapter.

4.2.8 Power Domains

The test chip design contains two separate core power domains. The primary goal of the separated power domain is to ease reading out the memory-based PUFs which require a power cycle. This way, the main part of the test chip core containing the communication interface can stay active while some of the memory-based PUFs in the separate power domain are power-cycled to generate a new response. Besides this goal, the separate power domain is also convenient when performing reliability tests under varying supply voltages.

The separate power domain contains one SRAM PUF instance (out of four), one D flip-flop PUF instance (out of four), one latch PUF instance (out of four), and one buskeeper PUF instance (out of two). The separate power domain has independent supply voltage and ground pins. Moreover, all signal lines connecting the two power domains can be blocked, effectively electrically isolating the PUF instances in the separate power domain.

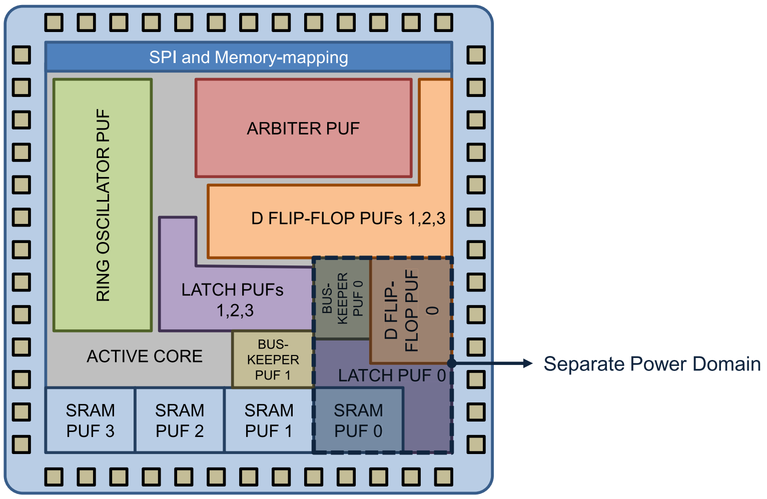


Fig. 4.3 Floor plan of the structures on the test chip

4.2.9 Implementation Details

Floor Plan

The schematic floor planning for the different PUF instances and other building blocks on the test chip's silicon die area is shown in Fig. 4.3. The separate power domain is also depicted, and as shown it contains one instance from all four memory-based PUFs. The active core is placed in the free space in between the different instances, in order to increase the impact of its toggling activity on the PUF evaluations.

Development Flow

Except for the arbiter and ring oscillator PUF blocks, the whole test chip design is described at the RTL level using a hardware description language (VHDL) and synthesized using a standard cell library. The back-end design is done by an external party (Invomec). The arbiter PUF is designed as a full-custom layout to have the most control over delay line and arbiter circuit balancing. The ring oscillator PUF is designed as an array of identically laid out hard-macro copies of an inverter chain. This is to make sure that all oscillators have the same nominal frequency, and any frequency difference is only caused by silicon process variations.

Implementation Technology

The final design of the test chip is implemented in 65 nm low-power CMOS technology (TSMC 65 nm CMOS Low Power MS/RF and TCBN65LP (nominal Vt)

Table 4.1 Silicon area breakdown of the different test chip building blocks

Building block	Silicon area (mm^2)	Relative area ($\cdot$/total logic)	Building block content
Ring Oscillator PUF	0.241	10.7 %	4096 ring oscillators + 16 × 32-bit counters + control
Latch PUF	0.272	9.5 %	4 × 8192 latches + 2 × multiplexer tree
SRAM PUF	0.213	12.1 %	4 × 64 kbit SRAM array
D Flip-Flop PUF	0.392	17.4 %	4 × 8192 D flip-flops + 2 × multiplexer tree
Arbiter PUF	0.279	12.4 %	256 × 64-bit arbiter PUF + control
Buskeeper PUF	0.076	3.4 %	2 × 8192 buskeeper cells + 2 × multiplexer tree
Active Core	0.353	15.7 %	32 × 128-bit substitution-permutation rounds
Additional Blocks	0.425	18.9 %	SPI interface, memory mapping, power control, ...
Total Logic Area	2.251	100.0 %	all of the above
Overhead	1.405	62.4 %	I/O pads, power/ground rings, empty space, ...
Complete Test Chip	3.656	162.4 %	1912 µm × 1912 µm silicon die

standard cell library). Both power domains of the core logic are nominally powered by $V_{dd} = 1.2$ V, and the I/O voltage is $V_{io} = 2.5$ V. The resulting silicon die is packaged in an LQFP64 package. In total, 192 packaged chips are produced.

Area Breakdown

Table 4.1 shows an estimate of the silicon die area breakdown of the different building blocks on the test chip. The estimates in this table are used in Chaps. 5 and 6 to estimate the required PUF size for a PUF-based application with given requirements. This provides an as objective as possible comparison between the different PUF constructions.

4.3 Experimental Uniqueness and Reproducibility Results

4.3.1 Evaluation of Delay-Based PUFs

Before we present the statistics of the experimental data, we first need to describe the manner in which we evaluate bit responses for the delay-based PUF constructions, i.e. the arbiter and the ring oscillator PUFs. The evaluation of the memory-based PUFs follows trivially from their design.

Arbiter PUF Evaluation Modes

The basic arbiter PUF already produces single bit responses. We also consider 2-XOR arbiter PUFs as proposed by Majzoobi et al. [94] by pairing up arbiter PUFs and perform an XOR-operation on their outputs to produce a single bit response. The XOR-operation is not implemented on the test chip but is performed off-line on the evaluated response bits of the basic arbiter PUFs.

Ring Oscillator PUF Evaluation Modes

As mentioned in Sect. 4.2.5, the ring oscillator PUF design outputs the frequency counter values directly. While these values already show some PUF behavior, it is difficult to use them as such in an application. For ease of integration, the frequency counter values need to be encoded in a meaningful binary response format. This will also make the comparison with the other PUF types more relevant. Ring oscillator PUF response bits are typically generated based on the relative comparisons between measured frequencies, as these comparisons are much more resilient to noise and varying evaluation conditions than the absolute frequency values. We present two encoding methods based on relative orderings of the frequency counter values.

The first method is a basic pairwise comparison (P.C.) between counted frequencies from different but simultaneously measured oscillators, as was proposed by Suh and Devadas [136] (but without the 1-out-of-k masking technique). A single response bit is generated based on the outcome of each comparison. To ensure independent responses, every oscillator is only used for the generation of a single bit. The arrangement of our ring oscillator PUF design in 16 batches, with 256 oscillators and one frequency counter per batch, allows us to measure 16 frequencies simultaneously and hence produce eight response bits in one evaluation. Using this evaluation method, the complete ring oscillator PUF design can generate $256 \times 8 = 2048$ response bits.

In [87], we developed a new response bit generation method for ring oscillator PUFs based on the ordering of the measured frequencies. Suh and Devadas [136] and Yin and Qu [157] observed that the amount of information in the ordering of n independent and identically distributed frequencies is as high as $\log_2 n!$. If one can find an efficient and noise-resilient encoding of such an ordering, this will lead to a significant increase in the number of independent response bits, since $\log_2 n! = \sum_{i=2}^{n} \log_2 i \approx n \cdot \log_2 \frac{n}{e}$ is superlinear in n, as opposed to the pairwise comparison method, which can only produce a number of response bits that is linear in n, i.e. $\frac{n}{2}$.

In [87], we propose using a Lehmer encoding [76, 125] to represent the ascending order of a vector of simultaneously measured frequencies, followed by a Gray encoding [44] of the Lehmer coefficients. A Lehmer code is a unique numerical representation of an ordering (permutation) which is moreover efficient to obtain since it does not require explicit value sorting. If $f^n = (f_1, \dots, f_n)$ is a vector containing n frequency measurements, then the Lehmer code of the ascending order of these values is a coefficient vector $r^{n-1} = (r_1, \dots, r_{n-1})$ with $r_i \in \{0, 1, \dots, i\}$. It is clear that r^{n-1} can take $2 \times 3 \times \cdots \times n = n!$ possible values, which is exactly the number of possible orderings of f^n; hence each ordering has a unique Lehmer code representation. The Lehmer coefficients are calculated from f^n as $r_j = \sum_{i=1}^{j} gt(f_{j+1}, f_i)$, with $gt(x, y) = 1$ if $x > y$ and 0 otherwise. The Lehmer encoding has the nice property that a minimal change in the sorted ordering, caused by two neighboring values swapping places after sorting, only changes a single Lehmer coefficient by ± 1. Using a binary Gray encoding for the Lehmer coefficients, this translates to only a single bit difference, which makes the overall response bit generation particularly noise-resilient. The length of the binary representation becomes $\sum_{i=2}^{n} \lceil \log_2 i \rceil$, which is a nearly optimal representation of the actual amount of information in the ordering, i.e. $\sum_{i=2}^{n} \log_2 i$.

For our ring oscillator PUF design on the test chip, we apply this Lehmer-Gray (L.G.) encoding off-line on each vector of $n = 16$ simultaneously measured frequencies, yielding 49 response bits. In total, the ring oscillator PUF can generate $256 \times 49 = 12544$ response bits using this method, which is over six times more than when using the pairwise comparison method.

4.3.2 PUF Experiment: Goals, Strategy and Setup

Experiment Goals

The goal of the experiments on the test chip is to characterize the meaningful properties of the eight studied PUF constructions as realistically and as accurately as possible. As explained in Sect. 4.1.1, it is not possible to make a comprehensive ranking of the different PUFs solely based on their bare characteristics. In order to make such an objective comparison, the envisioned application needs to be taken into account as well. We will do this respectively in Chaps. 5 and 6 for PUF-based authentication and PUF-based key generation. However, a common interface is desirable, i.e. a uniform set of characterization parameters for the meaningful properties of every PUF construction, which can be used in an unambiguous manner to determine their usability in a particular application. Here, we will provide such characterization parameters for the uniqueness and reproducibility of all studied PUF constructions, respectively in Sects. 4.3.3 and 4.3.4. The unpredictability of the different PUF constructions is discussed based on the response entropy in Sect. 4.4.

Experiment Strategy and Setup

To realistically determine the behavior of the different PUFs, in particular regarding their reproducibility, they need to be tested under varying conditions. In particular, we consider variations in test chip's supply voltage, $V_{dd} = 1.02\ \text{V} \dots 1.32\ \text{V}$, and environment temperature, $T_{env} = -45\ ^{\circ}\mathrm{C} \ldots 85\ ^{\circ}\mathrm{C}$, during evaluation. These conditions are created by powering the test chip with a variable power supply and placing it in a climate chamber with temperature control. To comprehensibly assess the PUF's behavior over these intervals, we test it at the four extreme corner conditions:

- The low-low corner (LL) or $\alpha_{\mathsf{LL}} = (T_{env} = -45\ ^{\circ}\mathrm{C}, V_{dd} = 1.02\ \mathrm{V})$.
- The low-high corner (LH) or $\alpha_{\mathsf{LH}} = (T_{env} = -45\ ^{\circ}\mathrm{C}, V_{dd} = 1.32\ \mathrm{V})$.
- The high-low corner (HL) or $\alpha_{\mathsf{HL}} = (T_{env} = 85\ ^{\circ}\mathrm{C}, V_{dd} = 1.02\ \mathrm{V})$.
- The high-high corner (HH) or $\alpha_{\mathsf{HH}} = (T_{env} = 85\ ^{\circ}\mathrm{C}, V_{dd} = 1.32\ \mathrm{V})$.

In addition, the test chip is also evaluated at the nominal reference condition: $\alpha_{\mathsf{ref}} = (T_{env} = 25\ ^{\circ}\mathrm{C}, V_{dd} = 1.20\ \mathrm{V})$.

At all considered conditions, all PUF constructions on all $N_{\mathsf{puf}} = 192$ test chips are evaluated for all their possible challenges, except for the arbiter PUFs, which are only evaluated on a set of 256 randomly generated challenges. In fact, for the memory-based PUFs, the notion of 'number of challenges' is rather arbitrary since it depends on how many bits one considers to be in a response, which we denote by ℓ_{resp}. In that respect, it is much more natural to detail the total number of generated response bits instead, denoted as $N_{\mathsf{bits}} \equiv N_{\mathsf{chal}} \times \ell_{\mathsf{resp}}$. Since all considered evaluation methods for the delay-based PUFs also generate bitwise responses, we also describe them in this manner. For the arbiter PUFs, we consider all bits generated by all arbiter PUFs at the same time; hence for the basic arbiter PUF experiment, $N_{\mathsf{bits}} = 256 \times 256 = 65536$, and for the 2-XOR arbiter PUF, $N_{\mathsf{bits}} = 256 \times 128 = 32768$. All response bits are evaluated $N_{\mathsf{meas}} = 20$ times under each condition for all PUFs on all chips.

4.3.3 Experimental PUF Uniqueness Results

Uniqueness Quantifiers

As expressed in Definition 8, the uniqueness of a PUF class is determined by the distribution of its inter-distance, in particular at nominal conditions. To study this distribution, we calculate the inter-distances $\mathbf{D}^{\mathsf{inter}}_{\mathsf{Exp}(\mathcal{P})}$ on the measured responses for all PUF constructions and report the most important statistics on the observed inter-distances in Table 4.2. Since all responses are bitwise, all inter-distances are measured using Hamming distances. However, to make it easier to compare the different PUF constructions, we report them as fractional Hamming distances, i.e. the Hamming distances are expressed as a ratio of the total number of evaluated response bits N_{bits}.

The basic inter-distance statistics we report in Table 4.2 are:

- The sample mean $\mu^{\mathsf{inter}}_{\mathcal{P}}$ and the sample standard deviation $\sigma^{\mathsf{inter}}_{\mathcal{P}}$, respectively expressing the location and dispersion of the inter-distance distribution.

Table 4.2 Experimental uniqueness results: inter-distance statistics (at nominal condition)

PUF	No.	N_{bits}	$\mu_{\mathcal{P}}^{\mathsf{inter}}$	$\sigma_{\mathcal{P}}^{\mathsf{inter}}$	$P[1\ \%]_{\mathcal{P}}^{\mathsf{inter}}$	$\min_{\mathcal{P}}^{\mathsf{inter}}$	$\hat{p}_{\mathcal{P}}^{\mathsf{inter}}$
SRAM PUF	0	65536	49.59 %	0.33 %	48.77 %	47.82 %	48.75 %
	1	65536	49.61 %	0.33 %	48.76 %	47.93 %	48.86 %
	2	65536	49.68 %	0.31 %	48.88 %	47.80 %	48.72 %
	3	65536	49.72 %	0.30 %	48.94 %	48.12 %	49.04 %
Latch PUF	0	8192	34.84 %	1.20 %	31.82 %	28.37 %	30.77 %
	1	8192	37.01 %	1.23 %	33.86 %	31.24 %	33.70 %
	2	8192	33.17 %	1.62 %	29.31 %	25.27 %	27.59 %
	3	8192	16.37 %	2.02 %	12.10 %	10.43 %	12.10 %
D Flip-Flop PUF	0	8192	42.35 %	0.83 %	40.36 %	38.14 %	40.70 %
	1	8192	42.42 %	1.01 %	40.15 %	38.20 %	40.76 %
	2	8192	41.88 %	0.89 %	39.80 %	37.87 %	40.43 %
	3	8192	41.20 %	0.87 %	39.15 %	36.95 %	39.50 %
Buskeeper PUF	0	8192	48.88 %	0.71 %	47.12 %	45.65 %	48.27 %
	1	8192	48.92 %	0.69 %	47.23 %	45.67 %	48.28 %
Arbiter PUF (basic)	0	65536	47.13 %	0.44 %	46.13 %	45.51 %	46.43 %
Arbiter PUF (2-XOR)	0	32768	49.74 %	0.29 %	49.07 %	48.40 %	49.71 %
Ring Oscillator PUF (P.C.)	0	2048	49.60 %	1.11 %	47.02 %	44.58 %	49.54 %
Ring Oscillator PUF (L.G.)	0	12544	46.86 %	0.48 %	45.77 %	44.34 %	46.45 %

- The first percentile $P[1\ \%]_{\mathcal{P}}^{\mathsf{inter}}$ of the samples and the sample minimum $\min_{\mathcal{P}}^{\mathsf{inter}}$. These two order statistics give a good idea of the left tail of the distribution, which is of interest when quantifying identifiability later.

In addition to these standard statistics, we introduce a custom statistic which will be of use later on: the inter-distance binomial probability estimator $\hat{p}_{\mathcal{P}}^{\mathsf{inter}}$. For many applications, we need to make assumptions about the distribution of the inter-distance of a PUF construction. In particular, we often need to extrapolate the observed empirical distribution to very small or large probabilities for which we have no reliable measurements. For efficiency reasons it is important that this be done as accurately as possible. However, any overestimation of the uniqueness could be disastrous for the security requirements of an application and should be avoided at all cost. For extrapolations beyond the observed inter-distance values, we make the assumption that the inter-distance is *binomially distributed* with parameter $\hat{p}_{\mathcal{P}}^{\mathsf{inter}}$. This parameter is chosen to be as large as possible, but sufficiently small such that all three following constraints are met, with $F_{\mathsf{bino}}(x; n, p)$ the cumulative binomial distribution function with parameters n and p, evaluated in x:

1. $F_{\mathsf{bino}}(\mu_{\mathcal{P}}^{\mathsf{inter}} \cdot N_{\mathsf{bits}}; N_{\mathsf{bits}}, \hat{p}_{\mathcal{P}}^{\mathsf{inter}}) \geq 50\ \%$, i.e. the estimated binomial distribution should produce values smaller than or equal to the observed sample mean with a probability of at least 50 %.
2. $F_{\mathsf{bino}}(P[1\ \%]_{\mathcal{P}}^{\mathsf{inter}} \cdot N_{\mathsf{bits}}; N_{\mathsf{bits}}, \hat{p}_{\mathcal{P}}^{\mathsf{inter}}) \geq 1\ \%$, i.e. the estimated binomial distribution should produce values smaller than or equal to the observed first percentile with a probability of at least 1 %. If this is not the case, values smaller than or equal to the first percentile of the samples are unlikely to occur as often as they do in the experiment, which means the estimated binomial distribution is an overestimation.
3. $F_{\mathsf{bino}}(\min_{\mathcal{P}}^{\mathsf{inter}} \cdot N_{\mathsf{bits}}; N_{\mathsf{bits}}, \hat{p}_{\mathcal{P}}^{\mathsf{inter}}) \geq 10^{-6}$, i.e. the estimated binomial distribution should produce values smaller than or equal to the observed sample minimum with a probability of at least 10^{-6} (the total number of observed samples is in the order of one million). If this is not the case, the observed sample minimum is unlikely to occur in the experiment according to the estimated binomial distribution, which is hence an overestimation.

The largest value for $\hat{p}_{\mathcal{P}}^{\mathsf{inter}}$ meeting all these three constraints is computed for all the PUF constructions and reported in Table 4.2. When the inter-distance is approximately binomially distributed, the value for $\hat{p}_{\mathcal{P}}^{\mathsf{inter}}$ should closely match that for $\mu_{\mathcal{P}}^{\mathsf{inter}}$. When this is not the case, the three constraints make sure that the binomial estimation based on $\hat{p}_{\mathcal{P}}^{\mathsf{inter}}$ at least accurately models the left tail of the actual inter-distance distribution, to avoid overestimation for extrapolations to small probabilities.

Discussion on Uniqueness Results

When presented as fractional Hamming distance, the optimal inter-distance is 50 %, which indicates two response bit vectors are maximally uncorrelated. A reported inter-distance sample mean (and binomial probability estimator) close to 50 % hence indicates high uniqueness. In this respect, the SRAM PUF and the buskeeper PUF perform particularly well, whereas the D flip-flip PUF and the latch PUF show slightly reduced uniqueness. In particular latch PUF instance number 3, with an average inter-distance of merely 16 %, shows strikingly little uniqueness, even in comparison to the other latch PUF instances. This is a sign of a possible implementation error in this latch instance. The basic arbiter PUF shows high uniqueness, which is an indication that the full-custom design approach succeeded in minimizing the arbiter PUF bias. For the 2-XOR arbiter PUF the uniqueness is evidently even higher. Both ring oscillator PUF evaluation methods also offer high uniqueness. The pairwise comparison approach has slightly better uniqueness than the Lehmer-Gray encoding, but the latter method of course produces significantly more response bits from the same amount of oscillators. For all the PUFs, the comparison between the inter-distance sample mean and binomial probability estimator is also an indication of how closely their inter-distance distribution resembles a binomial distribution.

4.3.4 Experimental PUF Reproducibility Results

Reproducibility Quantifiers

The reproducibility of a PUF class is determined by the distribution of its intra-distance (cf. Definition 7), in particular at the most extreme reference corner conditions. To study these distributions, we calculate the intra-distances $\mathbf{D}^{\mathsf{intra}}_{\mathsf{Exp}(\mathcal{P})}$ on the measured responses for all PUF constructions at the reference condition α_{ref}, and at all four corner conditions, α_{LL}, α_{LH}, α_{HL}, α_{HH}, and report the most important statistics on the observed intra-distances. Tables 4.3, 4.4, 4.5, 4.6 and 4.7 respectively report the intra-distance statistics for the experiments under conditions α_{ref}, α_{LL}, α_{LH}, α_{HL} and α_{HH}.

The basic intra-distance statistics we report in these tables are:

- The sample mean $\mu^{\mathsf{intra}}_{\mathcal{P}}$ and the sample standard deviation $\mu^{\mathsf{intra}}_{\mathcal{P}}$, respectively expressing the location and dispersion of the intra-distance distributions.
- The 99th percentile $P[99\ \%]^{\mathsf{intra}}_{\mathcal{P}}$ of the samples and the sample maximum $\max^{\mathsf{intra}}_{\mathcal{P}}$. These two order statistics give a good idea of the right tail of the distribution, which is of interest when quantifying identifiability later.

In addition to these standard statistics, we again introduce a custom statistic for realistically approximating the observed distribution by a binomial distribution: the intra-distance binomial probability estimator $\hat{p}^{\mathsf{intra}}_{\mathcal{P}}$. This time, this estimator is constrained to at least accurately model the right tail of the distribution to avoid underestimation of the intra-distance distribution at high values. The reported values for $\hat{p}^{\mathsf{intra}}_{\mathcal{P}}$ are computed to be as small as possible, but sufficiently large such that all three following constraints are met.

Table 4.3 Intra-distance statistics at nominal condition: $\alpha_{\mathsf{ref}} = (T_{env} = 25\,°\mathrm{C}, V_{dd} = 1.20\,\mathrm{V})$

PUF	No.	N_{bits}	$\mu_{\mathcal{P}}^{\mathsf{intra}}$	$\sigma_{\mathcal{P}}^{\mathsf{intra}}$	$P[99\,\%]_{\mathcal{P}}^{\mathsf{intra}}$	$\max_{\mathcal{P}}^{\mathsf{intra}}$	$\hat{p}_{\mathcal{P}}^{\mathsf{intra}}$
SRAM PUF	0	65536	5.46 %	0.14 %	5.77 %	6.06 %	5.63 %
	1	65536	5.46 %	0.14 %	5.76 %	6.02 %	5.59 %
	2	65536	5.46 %	0.14 %	5.76 %	5.96 %	5.55 %
	3	65536	5.47 %	0.14 %	5.76 %	6.00 %	5.58 %
Latch PUF	0	8192	2.61 %	0.24 %	3.16 %	3.65 %	2.77 %
	1	8192	2.78 %	0.25 %	3.37 %	3.92 %	3.00 %
	2	8192	3.40 %	0.34 %	4.28 %	5.16 %	4.10 %
	3	8192	2.64 %	0.55 %	4.13 %	5.27 %	4.20 %
D Flip-Flop PUF	0	8192	3.54 %	0.23 %	4.08 %	4.61 %	3.61 %
	1	8192	3.76 %	0.53 %	5.71 %	14.47 %	12.70 %
	2	8192	3.50 %	0.24 %	4.10 %	4.88 %	3.85 %
	3	8192	3.45 %	0.23 %	4.00 %	4.74 %	3.72 %
Buskeeper PUF	0	8192	4.16 %	0.24 %	4.72 %	5.21 %	4.22 %
	1	8192	4.17 %	0.24 %	4.74 %	5.21 %	4.23 %
Arbiter PUF (basic)	0	65536	3.04 %	0.08 %	3.23 %	3.39 %	3.07 %
Arbiter PUF (2-XOR)	0	32768	5.89 %	0.15 %	6.26 %	6.57 %	5.95 %
Ring Oscillator PUF (P.C.)	0	2048	1.53 %	0.39 %	2.44 %	3.13 %	1.80 %
Ring Oscillator PUF (L.G.)	0	12544	3.56 %	0.63 %	4.68 %	5.26 %	4.38 %

Table 4.4 Intra-distance statistics at LL corner conditions: $\alpha_{\mathsf{LL}} = (T_{env} = -40\,^{\circ}\text{C}, V_{dd} = 1.02\,\text{V})$

PUF	No.	N_{bits}	$\mu^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{LL}}}$	$\sigma^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{LL}}}$	$P[99\,\%]^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{LL}}}$	$\max^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{LL}}}$	$\hat{p}^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{LL}}}$
SRAM PUF	0	65536	7.36 %	0.19 %	7.79 %	8.02 %	7.55 %
	1	65536	7.33 %	0.19 %	7.81 %	8.11 %	7.62 %
	2	65536	7.33 %	0.19 %	7.80 %	8.09 %	7.60 %
	3	65536	7.34 %	0.19 %	7.83 %	8.11 %	7.62 %
Latch PUF	0	8192	23.10 %	1.92 %	26.58 %	28.21 %	25.91 %
	1	8192	23.36 %	1.73 %	26.89 %	28.35 %	26.04 %
	2	8192	15.85 %	1.11 %	18.73 %	19.58 %	17.76 %
	3	8192	9.34 %	1.56 %	14.62 %	17.11 %	15.22 %
D Flip-Flop PUF	0	8192	12.79 %	0.91 %	15.26 %	16.61 %	14.74 %
	1	8192	15.61 %	4.06 %	29.60 %	32.83 %	30.41 %
	2	8192	12.56 %	1.10 %	15.53 %	16.70 %	14.82 %
	3	8192	12.35 %	1.20 %	16.48 %	17.86 %	15.93 %
Buskeeper PUF	0	8192	9.68 %	0.41 %	10.62 %	11.39 %	9.86 %
	1	8192	9.89 %	0.39 %	10.80 %	11.44 %	10.04 %
Arbiter PUF (basic)	0	65536	7.41 %	0.23 %	7.98 %	8.25 %	7.75 %
Arbiter PUF (2-XOR)	0	32768	13.72 %	0.40 %	14.70 %	15.14 %	14.25 %
Ring Oscillator PUF (P.C.)	0	2048	3.75 %	0.47 %	4.83 %	5.62 %	3.88 %
Ring Oscillator PUF (L.G.)	0	12544	9.01 %	0.47 %	10.35 %	11.17 %	9.89 %

Table 4.5 Intra-distance statistics at LH corner conditions: $\alpha_{\mathsf{LH}} = (T_{env} = -40\ °\mathrm{C}, V_{dd} = 1.32\ \mathrm{V})$

PUF	No.	N_{bits}	$\mu^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{LH}}}$	$\sigma^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{LH}}}$	$P[99\ \%]^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{LH}}}$	$\max^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{LH}}}$	$\hat{p}^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{LH}}}$
SRAM PUF	0	65536	7.46 %	0.20 %	7.91 %	8.15 %	7.67 %
	1	65536	7.44 %	0.20 %	7.94 %	8.28 %	7.78 %
	2	65536	7.44 %	0.20 %	7.93 %	8.28 %	7.78 %
	3	65536	7.44 %	0.20 %	7.96 %	8.25 %	7.75 %
Latch PUF	0	8192	23.38 %	1.92 %	26.92 %	28.13 %	25.82 %
	1	8192	23.66 %	1.74 %	27.27 %	28.65 %	26.33 %
	2	8192	15.51 %	0.99 %	17.96 %	19.20 %	17.21 %
	3	8192	13.70 %	2.34 %	19.81 %	24.04 %	21.86 %
D Flip-Flop PUF	0	8192	12.90 %	0.91 %	15.33 %	16.92 %	15.03 %
	1	8192	15.68 %	4.08 %	29.63 %	33.02 %	30.60 %
	2	8192	12.64 %	1.09 %	15.59 %	16.75 %	14.87 %
	3	8192	12.42 %	1.20 %	16.52 %	17.76 %	15.83 %
Buskeeper PUF	0	8192	9.77 %	0.41 %	10.72 %	11.51 %	9.96 %
	1	8192	10.02 %	0.39 %	10.93 %	11.65 %	10.16 %
Arbiter PUF (basic)	0	65536	5.41 %	0.22 %	6.20 %	6.94 %	6.48 %
Arbiter PUF (2-XOR)	0	32768	10.23 %	0.39 %	11.58 %	12.88 %	12.02 %
Ring Oscillator PUF (P.C.)	0	2048	3.03 %	0.42 %	4.05 %	4.98 %	3.19 %
Ring Oscillator PUF (L.G.)	0	12544	7.81 %	0.41 %	8.69 %	9.32 %	8.15 %

Table 4.6 Intra-distance statistics at HL corner conditions: $\alpha_{\mathsf{HL}} = (T_{env} = 85\,°\text{C}, V_{dd} = 1.02\text{ V})$

PUF	No.	N_{bits}	$\mu^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{HL}}}$	$\sigma^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{HL}}}$	$P[99\,\%]^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{HL}}}$	$\max^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{HL}}}$	$\hat{p}^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{HL}}}$
SRAM PUF	0	65536	7.28 %	0.15 %	7.63 %	7.95 %	7.46 %
	1	65536	7.33 %	0.15 %	7.70 %	8.01 %	7.52 %
	2	65536	7.33 %	0.16 %	7.72 %	7.99 %	7.50 %
	3	65536	7.34 %	0.14 %	7.68 %	7.99 %	7.50 %
Latch PUF	0	8192	10.62 %	1.10 %	14.47 %	17.96 %	16.02 %
	1	8192	11.99 %	1.24 %	15.97 %	19.43 %	17.43 %
	2	8192	12.51 %	1.20 %	15.66 %	17.04 %	15.15 %
	3	8192	7.90 %	1.45 %	12.42 %	13.94 %	12.21 %
D Flip-Flop PUF	0	8192	18.10 %	0.79 %	20.11 %	21.03 %	19.11 %
	1	8192	17.95 %	1.66 %	21.89 %	23.71 %	21.54 %
	2	8192	18.27 %	0.78 %	20.23 %	21.08 %	19.23 %
	3	8192	18.15 %	0.81 %	19.96 %	20.84 %	18.96 %
Buskeeper PUF	0	8192	17.71 %	0.90 %	20.06 %	21.14 %	19.07 %
	1	8192	17.60 %	0.91 %	19.95 %	21.03 %	18.97 %
Arbiter PUF (basic)	0	65536	5.23 %	0.21 %	5.91 %	6.29 %	5.86 %
Arbiter PUF (2-XOR)	0	32768	9.90 %	0.37 %	11.13 %	11.83 %	11.00 %
Ring Oscillator PUF (P.C.)	0	2048	2.84 %	0.40 %	3.81 %	4.54 %	2.98 %
Ring Oscillator PUF (L.G.)	0	12544	7.11 %	0.42 %	8.10 %	9.19 %	8.03 %

Table 4.7 Intra-distance statistics at HH corner conditions: $\alpha_{\mathsf{HH}} = (T_{env} = 85\ °\mathrm{C}, V_{dd} = 1.32\ \mathrm{V})$

PUF	No.	N_{bits}	$\mu^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{HH}}}$	$\sigma^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{HH}}}$	$P[99\ \%]^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{HH}}}$	$\max^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{HH}}}$	$\hat{p}^{\mathsf{intra}}_{\mathcal{P};\alpha_{\mathsf{HH}}}$
SRAM PUF	0	65536	7.28 %	0.15 %	7.63 %	7.87 %	7.39 %
	1	65536	7.33 %	0.15 %	7.69 %	8.00 %	7.51 %
	2	65536	7.32 %	0.15 %	7.70 %	7.95 %	7.47 %
	3	65536	7.33 %	0.14 %	7.66 %	8.08 %	7.59 %
Latch PUF	0	8192	10.60 %	1.08 %	14.42 %	17.81 %	15.88 %
	1	8192	11.98 %	1.21 %	15.86 %	19.07 %	17.08 %
	2	8192	12.71 %	1.27 %	16.08 %	19.10 %	17.12 %
	3	8192	7.37 %	1.28 %	10.82 %	12.04 %	10.42 %
D Flip-Flop PUF	0	8192	17.89 %	0.79 %	19.90 %	20.87 %	18.90 %
	1	8192	17.77 %	1.67 %	21.79 %	23.38 %	21.22 %
	2	8192	18.12 %	0.78 %	20.06 %	20.84 %	19.06 %
	3	8192	17.98 %	0.81 %	19.81 %	20.48 %	18.82 %
Buskeeper PUF	0	8192	17.48 %	0.89 %	19.81 %	20.70 %	18.82 %
	1	8192	17.38 %	0.89 %	19.69 %	20.79 %	18.73 %
Arbiter PUF (basic)	0	65536	5.34 %	0.24 %	5.92 %	6.46 %	6.02 %
Arbiter PUF (2-XOR)	0	32768	10.11 %	0.44 %	11.14 %	12.13 %	11.30 %
Ring Oscillator PUF (P.C.)	0	2048	3.27 %	0.42 %	4.30 %	5.22 %	3.41 %
Ring Oscillator PUF (L.G.)	0	12544	8.35 %	0.53 %	9.89 %	10.79 %	9.54 %

1. $F_{\text{bino}}(\mu_{\mathcal{P}}^{\text{intra}} \cdot N_{\text{bits}}; N_{\text{bits}}, \hat{p}_{\mathcal{P}}^{\text{intra}}) \leq 50~\%$, i.e. the estimated binomial distribution should produce values larger than or equal to the observed sample mean with a probability of at least 50 %.
2. $F_{\text{bino}}(P[99~\%]_{\mathcal{P}}^{\text{intra}} \cdot N_{\text{bits}}; N_{\text{bits}}, \hat{p}_{\mathcal{P}}^{\text{intra}}) \leq 99~\%$, i.e. the estimated binomial distribution should produce values larger than or equal to the observed 99th percentile with a probability of at least 1 %. If this is not the case, values larger than or equal to the 99th percentile of the samples are unlikely to occur as often as they do in the experiment, which means the estimated binomial distribution is an underestimation.
3. $F_{\text{bino}}(\max_{\mathcal{P}}^{\text{intra}} \cdot N_{\text{bits}}; N_{\text{bits}}, \hat{p}_{\mathcal{P}}^{\text{intra}}) \leq 1-10^{-6}$, i.e. the estimated binomial distribution should produce values larger than or equal to the observed sample maximum with a probability of at least 10^{-6} (the number of samples in the experiment is in the order of one million). If this is not the case, the observed sample maximum is unlikely to occur in the experiment according to the estimated binomial distribution, which is hence an underestimation.

When the intra-distance is approximately binomially distributed, the value for $\hat{p}_{\mathcal{P}}^{\text{intra}}$ should closely match that for $\mu_{\mathcal{P}}^{\text{intra}}$. When this is not the case, the three constraints make sure that the binomial estimation based on $\hat{p}_{\mathcal{P}}^{\text{intra}}$ at least accurately models the right tail of the actual intra-distance distribution, to avoid underestimation for extrapolations to larger probabilities.

Discussion on Reproducibility Results

The intra-distance statistics reported in Tables 4.3 to 4.7 clearly show that the evaluation conditions impact the reproducibility of a PUF. For integration in a realistic application, we are particularly interested in the worst-case reproducibility behavior over all considered conditions. In Table 4.8, we summarize the worst-case results from Tables 4.3 to 4.7, i.e. the statistics from these tables which show the largest intra-distances. Note that different worst-case results can come from different conditions.

Studying the worst-case intra-distance statistics in Table 4.8, we see that of all the memory-based PUF constructions the SRAM PUF is particularly reproducible, with even worst-case maximal intra-distances smaller than 10 %. The buskeeper, D flip-flop and latch PUFs have a considerably worse reproducibility. We also remark two outlying behaviors:

1. Latch PUF instances numbers 2 and 3 show lower intra-distances than the other two, which is a side effect of them already having low uniqueness. This is most likely a result of an implementation fault in these instances, which we also already spotted based on the outlying uniqueness results. Note that latch instances numbers 2 and 3 deploy the rather complex scan chain-based read-out technique, as opposed to the other two which use a multiplexer tree, which is likely the cause of this problem.

Table 4.8 Worst-case intra-distance statistics over all four corners (α_{WC})

PUF	No.	N_{bits}	$\mu^{\mathrm{intra}}_{\mathcal{P};\alpha_{\mathrm{WC}}}$	$P[99\,\%]^{\mathrm{intra}}_{\mathcal{P};\alpha_{\mathrm{WC}}}$	$\max^{\mathrm{intra}}_{\mathcal{P};\alpha_{\mathrm{WC}}}$	$\hat{p}^{\mathrm{intra}}_{\mathcal{P};\alpha_{\mathrm{WC}}}$
SRAM PUF	0	65536	7.46 %	7.91 %	8.15 %	7.67 %
	1	65536	7.44 %	7.94 %	8.28 %	7.78 %
	2	65536	7.44 %	7.93 %	8.28 %	7.78 %
	3	65536	7.44 %	7.96 %	8.25 %	7.75 %
Latch PUF	0	8192	23.38 %	26.92 %	28.21 %	25.91 %
	1	8192	23.66 %	27.27 %	28.65 %	26.33 %
	2	8192	15.85 %	18.73 %	19.58 %	17.76 %
	3	8192	13.70 %	19.81 %	24.04 %	21.86 %
D Flip-Flop PUF	0	8192	18.10 %	20.11 %	21.03 %	19.11 %
	1	8192	17.95 %	29.63 %	33.02 %	30.60 %
	2	8192	18.27 %	20.23 %	21.08 %	19.23 %
	3	8192	18.15 %	19.96 %	20.84 %	18.96 %
Buskeeper PUF	0	8192	17.71 %	20.06 %	21.14 %	19.07 %
	1	8192	17.60 %	19.95 %	21.03 %	18.97 %
Arbiter PUF (basic)	0	65536	7.41 %	7.98 %	8.25 %	7.75 %
Arbiter PUF (2-XOR)	0	32768	13.72 %	14.70 %	15.14 %	14.25 %
Ring Oscillator PUF (P.C.)	0	2048	3.75 %	4.83 %	5.62 %	3.88 %
Ring Oscillator PUF (L.G.)	0	12544	9.01 %	10.35 %	11.17 %	9.89 %

2. D flip-flop PUF instance number one shows considerably larger intra-distances than the other three instances. The cause for this is unknown, but is likely also due to an implementation fault.

To prevent these outlying results from affecting this objective comparison between different PUF constructions, we will ignore them in all following analysises.

The delay-based PUFs also show relatively good reproducibility. The worst-case statistics of the basic arbiter PUF are nearly identical to those of the SRAM PUF. For the 2-XOR arbiter PUF, the intra-distances get approximately twice as large, which follows from their construction: when one of the two XOR-ed arbiter PUFs produces a faulty bit, the XOR-ed result will also be wrong. The ring oscillator PUF response bits based on pairwise comparison of frequencies are extremely reproducible, which proves the strength of this method. The Lehmer-Gray encoding method has worse reproducibility than the pairwise comparison method, but is still fairly good with a worst-case average smaller than 10 %.

4.4 Assessing Entropy

For many applications, and in particular for PUF-based key generation, it is important to accurately estimate the *entropy* of a random PUF response. Entropy is a function of the distribution of a random variable and expresses the amount of uncertainty one has about the outcome of the random variable. In the case of PUF responses, it represents a generalized and unconditional upper bound on the average predictability of an unobserved random outcome Y of a response evaluation. However, in general it is also very difficult or even impossible to calculate the entropy of a PUF response exactly. In the end, the distribution of most PUF responses is determined by very complex and even chaotic physical processes, and it cannot be learned in the complete detail which is required to calculate its entropy exactly. Typically, only estimated upper bounds on the underlying entropy can be provided. These bounds are either derived from a high-level physical model of the PUF construction, or based on the experimental data one observes.

In this section, we will present increasingly tighter upper bounds on the entropy of a PUF response based on increasingly more powerful adversary models, i.e. adversaries which gain more and more insight into the underlying distribution of the PUF's responses. We do this for all eight PUF constructions studied on the test chip and we compute entropy bounds based on the experimentally observed response distributions of these PUFs.

4.4.1 Adversary Models and Basic Entropy Bounds

In the following, Y^n represents a random bit vector of length n, and $Y_i \leftarrow \{0,1\}$ is a binary random variable whose distribution is completely determined by

$p_i \triangleq \Pr(Y_i = 1)$. We also use the following notation for the conditional distribution of Y_i conditioned on the previous bits $Y^{(i-1)} = (Y_1, \ldots, Y_{(i-1)})$: $p_{i|y^{(i-1)}} \triangleq \Pr(Y_i = 1|Y^{(i-1)} = y^{(i-1)})$. An overview of the general notions of probability theory and information theory used in this section is found in Appendix A.

Completely Ignorant Adversary

An adversary which is completely ignorant of the underlying distribution of the responses can make no better prediction than just guessing every bit completely at random. To him, it looks as if the PUF response has full entropy. Based on this adversary, we can introduce the following trivial response entropy bound:

$$H(Y^n) \leq n.$$

Using entropy density, this bound is denoted as

$$\rho_{\mathsf{ignorant}}(Y^n) \triangleq 100\ \%,$$

such that $\rho(Y^n) \leq \rho_{\mathsf{ignorant}}(Y^n)$. This *ignorant entropy bound* is very trivial, but we include it nonetheless for completeness and to detail the manner in which we will discuss the following bounds.

Adversary Knows Global Bias

The most basic deviation from a completely uniform and independent distribution of the response bits is caused by an overall global bias, i.e. on average every bit is more likely to be either ‘0’ or ‘1’. Such a global bias in the response Y^n can be expressed as:

$$p_{\mathsf{globalbias}} = \frac{1}{n}\mathsf{E}\left[\sum_{i=1}^{n} Y_i\right].$$

An adversary with knowledge of this global bias can make better than random predictions by guessing in favor of the bias, i.e. if $p_{\mathsf{globalbias}} < 50\ \%$ it predicts a ‘0’ and else a ‘1’. To the adversary, it looks as if all PUF response bits are independent and identically distributed (i.i.d.) according to a Bernoulli distribution with parameter $p_{\mathsf{globalbias}}$. Based on such an adversary, the following response entropy bound is introduced:

$$H(Y^n) \leq n \cdot h(p_{\mathsf{globalbias}}).$$

This *global bias entropy bound* is expressed using entropy density as $\rho(Y^n) \leq \rho_{\mathsf{globalbias}}(Y^n)$, with:

$$\rho_{\mathsf{globalbias}}(Y^n) \triangleq h(p_{\mathsf{globalbias}}).$$

Adversary Knows Bit-Dependent Bias

In a more realistic setting, every bit position in a PUF response vector will have its own bias, as expressed by $p_i = \Pr(y_i = 1)$. An adversary knowing these individual bit-dependent biases can make a more accurate prediction by guessing individual bits in favor of these biases. To the adversary, it looks as if all PUF response bits are independently, but no longer identically distributed, with each bit sampled from its own Bernoulli distribution with parameter p_i. Taking into account this adversary, the response entropy bound can be refined further:

$$H(Y^n) \leq \sum_{i=1}^{n} h(p_i).$$

Again we rewrite this *bit-dependent bias entropy bound* using entropy density as $\rho(Y^n) \leq \rho_{\mathsf{bitbias}}(Y^n)$, with:

$$\rho_{\mathsf{bitbias}}(Y^n) \triangleq \frac{1}{n}\sum_{i=1}^{n} h(p_i).$$

Adversary Knows Inter-Bit Dependencies

In the previous three adversary models, we moved from an adversary that sees a PUF response as an i.i.d. uniformly random bit vector to one that observes it as a vector of independently distributed bits which are no longer uniform or identically distributed. The next improvement to the adversary model would be to give it insight into the dependencies between different response bits, i.e. it no longer assumes that the response bits are completely independently distributed. It is clear that full knowledge of all inter-bit dependencies is generally unattainable since that would give the complete and exact distribution of the responses. Instead, we assume an adversary which has a certain realistic yet only partial model of the inter-bit dependencies. We consider two such partial dependency models:

1. The adversary has insight into the pairwise joint distributions $p(y_i, y_j)$ of all possible pairs of response bits in Y^n. This is a natural extension of insight into the bit-dependent bias of individual bits.
2. The adversary has partial insight into the conditional distribution $p(y_i|y^{(i-1)})$ of a response bit Y_i given the observation of the previous response bits $Y^{(i-1)} = y^{(i-1)}$. This is typically the case when the adversary deploys a successful next-bit modeling attack on response bits, with a prediction model which is trained on earlier observed responses.

We discuss both these adversarial models separately.

Adversary Knows Pairwise Joint Distributions

When an adversary knows all pairwise joint distributions between the response bits in Y^n, the response entropy bound is further lowered to:

$$H(Y^n) \leq \sum_{i=1}^{n} h(p_i) - \sum_{i=1}^{n-1} I(Y_i; Y_{i+1}).$$

We call this the *pairwise joint distribution entropy bound* and using entropy density, we write $\rho(Y^n) \leq \rho_{\mathsf{pairjoint}}(Y^n)$, with:

$$\rho_{\mathsf{pairjoint}}(Y^n) \stackrel{\triangle}{=} \frac{1}{n}\left(\sum_{i=1}^{n} h(p_i) - \sum_{i=1}^{n-1} I(Y_i; Y_{i+1})\right).$$

The mutual information values, i.e. the information shared by consecutive pairs of random bits, are subtracted from the bit-dependent bias entropy bound since they are in a way counted twice. The mutual information between two consecutive random bits can be computed from their pairwise joint distribution. Note that the subtracted amount is dependent on the way the individual random bit variables are ordered in Y^n, since the mutual information is computed over consecutive pairs (Y_i, Y_{i+1}) from $Y^n = (Y_1, \ldots, Y_n)$. Without loss of generalization, we assume that the bits are ordered in such a way as to maximize the subtracted amount. This yields the tightest lower bound on the response entropy.

Adversary Deploys Next-Bit Modeling Attack

In this model, we assume an adversary can perform a modeling attack on the PUF response bits, which after having been trained with $(i - 1)$ previously observed response bits, can predict the i-th bit with an average success probability of $p_{\mathsf{model}(i)}$. This results in a response entropy bound of:

$$H(Y^n) \leq \sum_{i=1}^{n} h(p_{\mathsf{model}(i)}).$$

We call this the *model entropy bound*, and in terms of entropy density this becomes $\rho(Y^n) \leq \rho_{\mathsf{model}}(Y^n)$, with:

$$\rho_{\mathsf{model}}(Y^n) \stackrel{\triangle}{=} \frac{1}{n}\sum_{i=1}^{n} h(p_{\mathsf{model}(i)}).$$

The tightness of this bound depends on the strength of the assumed model, and hence on the information and computational power available to the adversary in order to build this model. The goal of the model is to exploit dependencies between bits in order to make a better than random prediction for the next bit. In general, the model's success rate gets better and better as it is trained on more responses, i.e. $p_{\mathsf{model}(i)}$ increases with i. This also means that $\rho_{\mathsf{model}}(Y^n)$, unlike for the previously discussed bounds, is not a constant, but is dependent on n. In general, $\rho_{\mathsf{model}}(Y^n)$ decreases for increasing n. For example, if a model produces near-perfect predictions after having been trained with a large number of observed response bits, all following response bits will no longer contribute any meaningful entropy since they are perfectly predictable. Hence, producing more response bits will only increase the response length and not the entropy, i.e. the entropy density of the response decreases.

4.4.2 Entropy-Bound Estimations Based on Experimental Results

Next, we evaluate the different entropy bounds on the measured responses obtained from the performed experiments which were discussed in Sect. 4.3. All entropy bounds are expressed using entropy density, allowing us to make an easy comparison between different PUF constructions.

Bound Estimation Strategy

For all memory-based PUF instances, we consider a response bit vector containing $n = 5000$ bits to estimate the following entropy bounds:

- The ignorant entropy bound is trivially equal to 100 % for all PUFs.
- For the global bias entropy bound, we estimate the global bias by computing the response bit sample mean over all 5000 considered bits on all 192 devices.
- For the bit-dependent bias entropy bound, we estimate the bit-dependent biases by computing the response bit sample mean over all 192 devices for every bit individually.
- For the pairwise distribution entropy bound, we estimate the pairwise joint distributions of all possible pairs of the considered response bits, by counting the occurrences of each of the four possible outcomes $(0, 0)$, $(0, 1)$, $(1, 0)$ or $(1, 1)$ for each considered pair on all 192 devices.

We do not consider the model entropy bound for memory-based PUFs, as no successful modeling attacks on memory-based PUFs are known. All bound estimations are done based on responses measured at nominal conditions.

For the delay-based PUF instances, we differentiate between the arbiter and the ring oscillator PUFs. For both response generation methods of the ring oscillator PUF, we perform the same bound estimations as for the memory-based PUFs, only on different sizes of response bit vectors, respectively $n = 2048$ for the pairwise comparison method and $n = 12544$ for the Lehmer-Gray method. No model-building attacks for these ring oscillator PUFs are known.

For the arbiter PUF we take a slightly different approach. The first four bounds are computed for response bit vectors of only $n = 256$ bits, for each of the 256 arbiter PUF instances and 128 2-XOR arbiter PUF instances on the test chip individually, but we report only the results for the worst observed instance. Besides these four bounds, we also consider the model entropy bound for both types of arbiter PUFs. This is discussed in more detail in Sect. 4.4.3.

Bound Estimation Results

The results for the estimations of the ignorant entropy bound, the global bias entropy bound, the bit-dependent bias entropy bound and the pairwise distribution entropy bound are presented in Table 4.9. From these results, it is evident that these four estimates represent consecutively tighter upper bounds on the real response entropy.

4.4.3 Modeling Attacks on Arbiter PUFs

From the initial introduction of arbiter PUFs, it was recognized that they are susceptible to modeling attacks. This is a result of the reduced complexity of the dependency between arbiter PUF challenges and responses. For the basic arbiter PUF it was made clear, e.g. by Lee et al. [75], that this dependency is to a high level of accuracy even a linear system. In that case, the unpredictability of the responses results only from the unknown parameters of this underlying system, since once they are learned every response bit is easily predictable with high accuracy. A modeling attack attempts to estimate the unknown model parameters as a function of observed challenge-response pairs. For more advanced arbiter PUF constructions, e.g. the 2-XOR arbiter PUF, the underlying model becomes more complex, but as shown by Rührmair et al. [118], it is still modelable with more advanced modeling techniques.

Modeling with Machine Learning Techniques

A particularly interesting set of modeling techniques are based on machine learning [99]. Machine learning algorithms are able to automatically learn complex behavior and unknown model parameters by generalizing on presented training examples. Another strong motivation for using machine learning algorithms in modeling attacks is that they are generic, i.e. they have the ability to learn any complex behavior and are not a priori restricted to a particular model description (e.g. a linear model).

In [55], we apply two basic machine learning techniques to our experimental arbiter PUF results to test for modelability: (i) artificial neural networks or ANNs [99],

Table 4.9 Entropy density upper-bound estimations

PUF	No.	n	$\rho_{\text{ignorant}}(Y^n)$	$\rho_{\text{globalbias}}(Y^n)$	$\rho_{\text{bitbias}}(Y^n)$	$\rho_{\text{pairjoint}}(Y^n)$
SRAM PUF	0	5000	100.00 %	99.99 %	99.04 %	94.11 %
	1	5000	100.00 %	99.99 %	99.05 %	94.09 %
	2	5000	100.00 %	99.99 %	99.14 %	94.18 %
	3	5000	100.00 %	99.99 %	99.20 %	94.27 %
Latch PUF	0	5000	100.00 %	80.01 %	77.12 %	71.92 %
	1	5000	100.00 %	81.69 %	79.56 %	74.36 %
	2	5000	100.00 %	91.88 %	90.91 %	85.84 %
	3	5000	100.00 %	59.22 %	51.05 %	45.51 %
D Flip-Flop PUF	0	5000	100.00 %	88.88 %	88.36 %	83.36 %
	1	5000	100.00 %	89.13 %	88.65 %	83.70 %
	2	5000	100.00 %	88.00 %	87.55 %	82.57 %
	3	5000	100.00 %	86.77 %	86.34 %	81.34 %
Buskeeper PUF	0	5000	100.00 %	98.82 %	97.94 %	93.00 %
	1	5000	100.00 %	99.03 %	98.02 %	93.05 %
Arbiter PUF (basic)	0	256	100.00 %	94.27 %	92.79 %	89.62 %
Arbiter PUF (2-XOR)	0	256	100.00 %	99.39 %	98.72 %	95.82 %
Ring Oscillator PUF (P.C.)	0	2048	100.00 %	99.99 %	99.04 %	94.69 %
Ring Oscillator PUF (L.G.)	0	12544	100.00 %	99.92 %	94.87 %	86.63 %

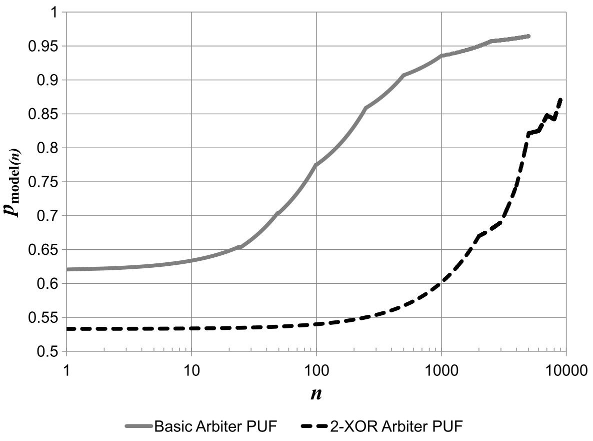


Fig. 4.4 Average success rate of a machine-learning attack on basic and 2-XOR arbiter PUFs. The success rate $p_{\text{model}(n)}$ presents the probability of correctly predicting the n-th response bit after having been trained with $(n-1)$ previously observed bits

and (ii) support-vector machines or SVMs [26]. Both techniques have been demonstrated to be able to effectively model basic arbiter PUFs, respectively by Gassend et al. [43] and Rührmair et al. [118]. However, we apply these attacks on PUF responses resulting from a modern implementation (65 nm CMOS), as opposed to these earlier results which work with older technologies [43] or only with simulated data [118]. For this analysis of the model entropy bound, we are mainly interested in the results of these modeling attacks. For more details on their implementation, we refer to [55].

Modeling Results

Using both ANN and SVM, we were able to successfully model both basic and 2-XOR arbiter PUFs. Both ANN and SVM first take a number of known arbiter PUF challenge-response pairs which they use to train their model. Afterwards, their modeling performance is evaluated by their success rate of accurately predicting unobserved responses when presented with a challenge from a large test set. It is evident that, the more training examples a machine learning algorithm is allowed to use, the better its modeling accuracy becomes. In Fig. 4.4, we summarize the outcome of our machine learning modeling attacks on the experimental data from the basic and the 2-XOR arbiter PUFs. It shows the average success rate ($p_{\text{model}(i)}$) of the best machine learning technique, ANN or SVM, after having been trained with $(i-1)$ earlier observed response bits.

We can draw some conclusions on the machine learning results as represented in Fig. 4.4:

- The 2-XOR arbiter PUF is more difficult to model with our techniques than the basic arbiter PUF, as expressed by the larger number of training examples required to achieve the same modeling success rate. This is a result of the challenge-response relation being more complex for the 2-XOR arbiter PUF.

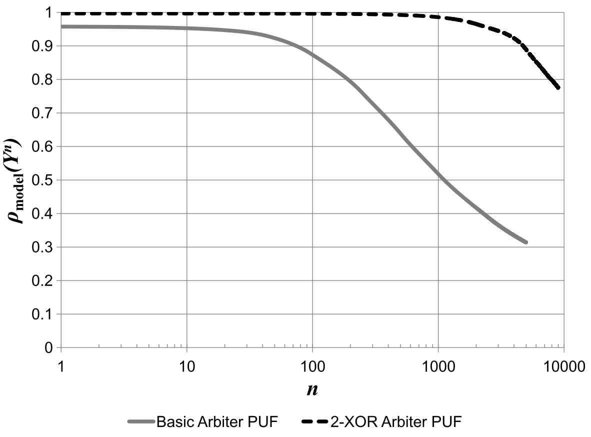


Fig. 4.5 Estimated entropy density bounds for the basic and 2-XOR arbiter PUFs, based on a modeling adversary deploying the machine-learning attack results presented in Fig. 4.4

- The basic arbiter PUF can be modeled with ≈90 % accuracy after training with ≈500 examples and with ≈95 % accuracy after training with ≈2000 examples. Note that the maximal attainable success rate of any modeling attack is naturally limited by the reproducibility of the considered PUF instance. In that respect, the obtained modeling accuracy of ≈97 % after training with ≈5000 examples can be considered perfect, given that the average intra-distance of the basic arbiter PUF at nominal conditions is about 3 % (cf. Table 4.3). This means that all following response bits do not contribute any entropy except for their reproducibility uncertainty.
- The 2-XOR arbiter PUF can be modeled with ≈75 % accuracy after training with ≈4000 examples and with nearly 90 % after training with 9000 examples.

Modeling Entropy Bound

Using the machine learning modeling attack results as presented in Fig. 4.4, we can calculate the model entropy bound as $\rho_{\text{model}}(Y^n) = \frac{1}{n}\sum_{i=1}^{n} h(p_{\text{model}(i)})$. The resulting entropy bound, as a function of n, is presented in Fig. 4.5.

From Fig. 4.5 we learn that for the basic arbiter PUF, $\rho_{\text{model}}(Y^{100}) < 90\ \%$ and $\rho_{\text{model}}(Y^{1000}) \approx 50\ \%$. For response bit vectors of more than 1000 bits, the entropy density drops drastically as each additional response bit adds very little entropy. After about 5000 bits, each additional response bit only adds a little noise entropy, which is not useful. For the 2-XOR PUF, the results are less severe with $\rho_{\text{model}}(Y^{5000} < 90\ \%)$ and $\rho_{\text{model}}(Y^{8000} < 80\ \%)$.

As a final remark, we want to make clear that the obtained model entropy bounds are likely not very tight, since our machine-learning modeling attacks are not optimal. More advanced or more fine-tuned modeling techniques are likely to obtain even higher success rates, and hence lower model entropy bounds, than our results, which are based on relatively basic machine learning algorithms.

4.5 Conclusion

In this chapter, we have presented a detailed discussion on the realization, evaluation and analysis of a collection of different PUF constructions in a realistic, practical and objective manner. We have developed and produced a test chip carrying six different intrinsic PUF implementations: four memory-based PUFs (SRAM, latch, D flip-flop and buskeeper) and two delay-based PUFs (arbiter and ring oscillator). Both delay-based PUFs are evaluated (off-line) using two different methods, giving a total of eight studied PUF constructions. A large-scale experimental evaluation is performed of all eight PUF constructions on 192 manufactured test chips under different temperature and supply voltage conditions. The large response data set produced by this experiment is meticulously analyzed in order to assess the behavior of the different PUF constructions, with regard to their reproducibility, their uniqueness and the entropy of their responses.

4.5.1 Summary of PUF Behavior Results

As a final conclusion, we present a concise summary of the most important results related to the PUF behavior of the different intrinsic PUF implementations which we studied in this chapter. We characterize the different PUFs for each of the three analyzed properties (reproducibility, uniqueness and response entropy) in a single quantifier which will be of particular practical use in the following chapters, which discuss PUF-based applications. The three quantifiers we consider are:

1. The intra-distance binomial probability estimator $\hat{p}_{\mathcal{P}}^{\mathsf{intra}}$ as a characterization of reproducibility. This will allow us to accurately and safely estimate the right tail of the intra-distance distribution, i.e. the probability that the intra-distance becomes very large.
2. The inter-distance binomial probability estimator $\hat{p}_{\mathcal{P}}^{\mathsf{inter}}$ as a characterization of uniqueness. This quantifier allows us to accurately and safely estimate the left tail of the inter-distance distribution, i.e. the probability that very low inter-distances occur.
3. The tightest upper bound on the response entropy density $\rho(Y^n)$ as a characterization of response entropy.

For all three quantifiers, we selected the worst-case measured values over all implemented instances of the individual PUF types, respectively from Tables 4.8, 4.2 and 4.9. Note that we discarded all results from latch PUF instances numbers 2 and 3 and from D flip-flop PUF instance number 1, since they exhibit a strong outlier behavior which is likely caused by an implementation error. For the entropy density bound on the basic and 2-XOR arbiter PUFs, we cannot provide a single constant quantifier, since their entropy density bound depends on the considered response length. Instead, we refer to Fig. 4.5, which shows this relation for the machine-learning modeling attacks we performed. Relating to the reported entropy density

Table 4.10 Summary of the most important results on the PUF behavior of the implemented intrinsic PUFs

PUF class $\mathcal{P}$	$\hat{p}_{\mathcal{P}}^{\text{intra}}$	$\hat{p}_{\mathcal{P}}^{\text{inter}}$	$\rho(Y^n)\leq$
SRAM PUF	7.78 %	48.72 %	94.09 %
Latch PUF	26.33 %	30.77 %	71.92 %
D Flip-Flop PUF	19.23 %	39.50 %	81.34 %
Buskeeper PUF	19.07 %	48.27 %	93.00 %
Arbiter PUF (basic)	7.75 %	46.43 %	Fig. 4.5
Arbiter PUF (2-XOR)	14.25 %	49.71 %	Fig. 4.5
Ring Oscillator PUF (P.C.)	3.88 %	49.54 %	94.69 %
Ring Oscillator PUF (L.G.)	9.89 %	46.45 %	86.63 %

results, we also point out that these are merely upper bounds and that the actual response entropy is smaller.

The summary of PUF behavior results in Table 4.10, together with the overview of the area breakdown of the different PUF implementations on the test chip presented in Table 4.1, will be of great value for assessing and optimizing the deployment of these PUF constructions in actual applications.