Chapter 2
Physically Unclonable Functions: Concept and Constructions

2.1 Introduction

2.1.1 The PUF Concept

Since it is the main subject of this book, it is important to clarify and define the basic concept of a physically unclonable function or PUF. However, for a variety of reasons, this task turns out to be less straightforward than expected. The collection of proposed constructions labelled 'PUF' is growing at such a pace, both in depth and breadth, that one can easily call it a *zoo* at this moment.[1] There is also an equally substantial group of constructions which could, but are for the moment not, called 'PUFs', among other reasons because they were proposed before the acronym 'PUF' had been coined, or because they were proposed by authors unfamiliar with the acronym, e.g. in fields outside hardware security engineering. All these differing constructions are moreover proposed in a greatly varying patchwork of implementation materials, technologies and platforms. Finding similar or even identifying properties which accurately capture what is understood to be a PUF is hence far from trivial. In this chapter and the next, we discuss the PUF concept in great detail in order to do exactly that: first, by extensively studying existing PUF constructions in Chap. 2, and next by describing, assessing and finally formalizing the observed properties of PUFs in Chap. 3.

On PUFs and Fingerprints

In an attempt to express the concept of a PUF in a single phrase, one of the best possible descriptions would be: "a PUF is an object's fingerprint". PUFs are similar to fingerprints for more than one reason:

[1] By analogy with the Complexity Zoo [139] and the SHA-3 Zoo [140].

R. Maes, *Physically Unclonable Functions*, DOI 10.1007/978-3-642-41395-7_2,

- A human fingerprint is a feature which strongly expresses *individualism*, i.e. it answers the question "Which human is this?" as opposed to other features such as having two eyes and ears and walking on two legs, which express *essentialism*, i.e. answering the question "What is a human?". In a more inanimate sense, a PUF bears the same meaning for a class of objects: a PUF is an identifying feature of a specific instance of a class of objects, or for short is an *instance-specific feature*.
- As an individualising feature, a fingerprint is also *inherent*, i.e. every human being is born with fingerprints, unlike other identifying qualities like a name or a written signature, which are bestowed upon or acquired by an individual after birth. In the same way, PUFs are inherently present in an object from its creation, as a result of unique variations during its creation process.
- Finally, as an inherent individualising feature, fingerprints are also *unclonable*, i.e. the physical and biological processes which determine a human being's fingerprints are beyond any level of meaningful control which would allow us to create a second individual with the same inherent fingerprints.[2] This even holds for human beings sharing the same genetic material, like identical twins. Hence even a (hypothetical) biological clone of a person would not share that person's fingerprints. In this aspect PUFs are very similar to fingerprints. In fact, by its being the central specifier in the term 'physically unclonable functions', (physical) unclonability is one of *the* core properties of a PUF.

Following this discussion, we propose the following colloquial dictionary definition of the PUF concept: "*a PUF is an expression of an inherent and unclonable instance-specific feature of a physical object*". By indicating that a PUF is always object-related, we explicitly distinguish PUFs from fingerprints and other biometric parameters which naturally reflect on human beings. However, as discussed above, in many aspects PUFs and biometrics are equivalent.

2.1.2 Chapter Goals

Order in the PUF Zoo

The amount of published results on physically unclonable functions has increased exponentially over the last few years, and the need for a synthesizing effort presents itself. This chapter presents an objective, large-scale and in-depth overview of the myriad of PUFs which have been proposed over the years. The main goals of this overview are:

- Present an as complete as possible reference list of PUF and PUF-like proposals to date, with a focus on so-called *intrinsic* PUFs.

[2] By unclonable we do not mean that it is impossible to create or obtain a facsimile of a person's fingerprints; in fact human beings create copies of their fingerprints every time they touch a smooth surface.

- Introduce and apply a classification of PUF constructions based on practical considerations.
- Present a common descriptive framework for assessing the basic functionality and quality of a PUF, and use it to produce an as fair as possible comparative quantitative analysis of different PUF implementations based on published experimental results.

Besides these clear objectives, this overview will also be of great aid in identifying interesting subclasses, properties, applications, and even open problems in the field of PUFs.

2.1.3 Chapter Overview

Before we start describing different PUF constructions and their qualities, we will in Sect. 2.2 introduce the basic nomenclature and notational conventions related to PUFs which we will use throughout this book. These conventions make it much easier to relate and compare identical concepts in often widely differing implementations and data sets, and it is therefore highly recommended you read this section first. In Sect. 2.3, we delve deeper into the semantics of the acronym 'PUF' and also discuss a number of possible classifications in the large variety of PUF constructions which have been proposed over time. The main body of this chapter is a very extensive overview of known PUF constructions,[3] detailing the implementation of each proposal and reporting experimental results, if any. In this book, we mainly focus on *intrinsic* PUF constructions which are discussed in Sect. 2.4. For more information on existing *non-intrinsic* PUF constructions we refer to Appendix B. In Sect. 2.5, a number of concepts are discussed which present extensions or modes of operation of PUFs. Finally, we conclude this chapter in Sect. 2.6.

2.2 Preliminaries

2.2.1 Conventions on Describing PUFs

Formally describing the operation of a PUF is a tedious yet important task. Due to the multitude of different PUF constructions and different ways of considering a PUF outcome, one quickly runs the risk of either introducing confusion due to an ambiguous description or losing important details due to a too sparse description. Here, we introduce the verbal and notational description of PUFs which we will use and build upon throughout the rest of this book. Since this is only intended as

[3] We aim at giving an exhaustive overview of PUF proposals to date, but with the plethora of new constructions appearing in recent years, it is very likely that some are missing.

a convenient but clear notation format, we refrain at this point as much as possible from making assumptions or putting restrictions on the underlying notions. Instead, we limit these conventions to the most basic concepts required to formally talk about PUFs. For completeness, the basic mathematical notations and definitions used in this section and the rest of this book are shortly introduced in Appendix A.

PUF Class

We first introduce the notion of a PUF class, denoted by $\mathcal{P}$, which is the complete description of a particular PUF construction type. A PUF class provides a creation procedure $\mathcal{P}.\mathsf{Create}$ which is invoked to create instances of $\mathcal{P}$. In general $\mathcal{P}.\mathsf{Create}$ is a probabilistic procedure, which we make explicit when necessary by providing it with a randomized input $\mathcal{P}.\mathsf{Create}(r^{\mathsf{C}})$, with $r^{\mathsf{C}} \xleftarrow{\$} \{0,1\}^*$ representing an undetermined number of fair coin tosses.

From a practical point of view, $\mathcal{P}$ represents the detailed structural design or *blueprint* of a PUF construction and $\mathcal{P}.\mathsf{Create}$ the detailed (physical) production process to build the design.

PUF Instance

A PUF instance puf is a discrete instantiation of a PUF class $\mathcal{P}$, as generated by its creation procedure $\mathcal{P}.\mathsf{Create}$. In all respects, a PUF class can be considered to be the set of all its created instances:

$$\mathcal{P} \equiv \left\{ \mathsf{puf}_i \leftarrow \mathcal{P}.\mathsf{Create}\left(r_i^{\mathsf{C}}\right) : \forall i, r_i^{\mathsf{C}} \xleftarrow{\$} \{0,1\}^* \right\}.$$

A PUF instance puf is considered as a particular crystallized *state* of the construction described by its PUF class $\mathcal{P}$. The construction of many PUF classes makes it possible to configure part of the PUF instance's state, i.e. it is not fixed by the creation procedure but can be set and easily modified by means of an external input. When required, we specify the configurable state x of a PUF instance puf as $\mathsf{puf}(x)$.

From a practical point of view, the *state* represented by a PUF instance puf is the exact (physical) structure of a produced PUF construction. The configurable part of a PUF instance's state is generally called the *challenge* which is *applied* to the PUF instance, and we will use the same terminology in this book. The set of all possible challenges x which can be applied to an instance of a PUF class $\mathcal{P}$ is denoted as $\mathcal{X}_{\mathcal{P}}$.

PUF Evaluation

Every PUF instance puf provides an evaluation procedure $\mathsf{puf}.\mathsf{Eval}$ which produces a quantitative outcome representing a measurement of the PUF instance. The outcome produced by $\mathsf{puf}.\mathsf{Eval}$ depends on the state represented by the PUF instance. When the PUF instance is *challengeable*, we write: $\mathsf{puf}(x).\mathsf{Eval}$. In general $\mathsf{puf}(x).\mathsf{Eval}$

is also a probabilistic procedure which we again make explicit when necessary as $\mathsf{puf}(x).\mathsf{Eval}(r^{\mathsf{E}} \xleftarrow{\$} \{0,1\}^*)$.

From a practical point of view, a PUF instance evaluation is the outcome of a (physical) experiment, generating a particular measurement of the PUF instance's physical state at that moment. Such a measurement is generally called the *response* of the PUF instance and we will use the same terminology. The class of all possible response values which a PUF instance of a PUF class $\mathcal{P}$ can produce is denoted as $\mathcal{Y}_{\mathcal{P}}$.

For many PUF classes, the outcome of a PUF instance evaluation is also affected by external physical parameters, e.g. environment temperature, supply voltage level, etc. We call this the *condition* of the evaluation. When required, we denote this as $\mathsf{puf}(x).\mathsf{Eval}^{\alpha}$, e.g. with $\alpha = (T_{env} = 80\ ^{\circ}\mathrm{C})$, meaning that this evaluation took place at an environment temperature of 80 °C. When the condition specifier α is omitted, one may assume an evaluation at nominal operating conditions.

Shorthand Notation

To avoid the rather verbose use of the randomization variables r^{C} and r^{E}, we introduce a more convenient and compact notation using random variables.

Creation of a random PUF instance:

$$\mathsf{puf}_i \leftarrow \mathcal{P}.\mathsf{Create}\big(r_i^{\mathsf{C}} \xleftarrow{\$} \{0,1\}^*\big) \text{ becomes}$$

$$\mathsf{PUF} \leftarrow \mathcal{P}.\mathsf{Create}, \text{ or the even shorter } \mathsf{PUF} \leftarrow \mathcal{P}.$$

Random evaluation of PUF instance puf_i on challenge x:

$$y_i^{(j)}(x) \leftarrow \mathsf{puf}_i(x).\mathsf{Eval}\big(r_j^{\mathsf{E}} \xleftarrow{\$} \{0,1\}^*\big) \text{ becomes}$$

$$Y_i(x) \leftarrow \mathsf{puf}_i(x).\mathsf{Eval}, \text{ or the even shorter } Y_i(x) \leftarrow \mathsf{puf}_i(x).$$

Random evaluation of a random PUF instance on challenge x:

$$Y(x) \leftarrow \mathsf{PUF}(x).\mathsf{Eval}, \text{ or the shorter } Y(x) \leftarrow \mathsf{PUF}(x).$$

2.2.2 Details of a PUF Experiment

A PUF response is generally considered a random variable. To assess the usability of a PUF class, information about the distribution of its PUF responses is needed. A possible way to obtain such knowledge is through experiment,[4] i.e. estimating distribution statistics from observed PUF response values.

[4] An alternative method of learning information about PUF response distributions is through physical modeling of the PUF class construction.

Definition and Parameters of a PUF Experiment

PUF response values observed in an experiment can be ordered in a number of different ways. Three important 'dimensions' in an array of observed PUF responses are discernable: (i) responses from different PUF instances; (ii) responses from the same PUF instance but on different challenges; and (iii) responses from the same PUF instance on the same challenge but from distinct evaluations.

Definition 1 (PUF Experiment) An $(N_{\mathsf{puf}}, N_{\mathsf{chal}}, N_{\mathsf{meas}})$-experiment on a PUF class $\mathcal{P}$ is an array of PUF response values obtained through observation. An $(N_{\mathsf{puf}}, N_{\mathsf{chal}}, N_{\mathsf{meas}})$-experiment contains $N_{\mathsf{puf}} \times N_{\mathsf{chal}} \times N_{\mathsf{meas}}$ values, consisting of response evaluations on N_{puf} (random) PUF instances from the considered PUF class $\mathcal{P}$, challenged on the same set of N_{chal} (random) challenges, with N_{meas} distinct response evaluations for each challenge on each PUF instance.

$$\mathsf{Experiment}_{\mathcal{P}}(N_{\mathsf{puf}}, N_{\mathsf{chal}}, N_{\mathsf{meas}}) \rightarrow \mathbf{Y}_{\mathsf{Exp}(\mathcal{P})} = \left[y_i^{(j)}(x_k) \leftarrow \mathsf{puf}_i(x_k).\mathsf{Eval}\left(r_j^{\mathsf{E}}\right) \right],$$

with

$$\forall 1 \leq i \leq N_{\mathsf{puf}} : \ \mathsf{puf}_i \xleftarrow{\$} \mathcal{P},$$

$$\forall 1 \leq k \leq N_{\mathsf{chal}} : \ x_k \xleftarrow{\$} \mathcal{X}_{\mathcal{P}},$$

$$\forall 1 \leq j \leq N_{\mathsf{meas}} : \ r_j^{\mathsf{E}} \xleftarrow{\$} \{0,1\}^*.$$

If the conditions of a PUF experiment are of importance, the condition specifier is mentioned as $\mathsf{Experiment}^{\alpha}_{\mathcal{P}}(N_{\mathsf{puf}}, N_{\mathsf{chal}}, N_{\mathsf{meas}})$, which expresses that all PUF responses of the experiment are evaluated under these conditions.

2.2.3 PUF Response Intra-distance

Intra-distance Definition

Definition 2 A PUF response **intra**-distance is a random variable describing the distance between two PUF responses from the same PUF instance and using the same challenge:

$$D^{\mathsf{intra}}_{\mathsf{puf}_i}(x) \triangleq \mathbf{dist}\left[Y_i(x); Y_i'(x) \right],$$

with $Y_i(x)$ and $Y_i'(x)$ two distinct and random evaluations of PUF instance puf_i on the same challenge x. Additionally, the PUF response intra-distance for a random PUF instance and a random challenge is defined as the random variable:

$$D^{\mathsf{intra}}_{\mathcal{P}} \triangleq D^{\mathsf{intra}}_{\mathsf{PUF} \leftarrow \mathcal{P}}(X \leftarrow \mathcal{X}_{\mathcal{P}}).$$

dist$[\cdot;\cdot]$ can be any well-defined and appropriate distance metric over the response set $\mathcal{Y}$. In this book, responses are nearly always considered as bit vectors and the distance metric used is the Hamming distance or the fractional Hamming distance (cf. Sect. A.1.1).

Intra-distance Statistics

For the design of nearly all PUF-based applications, knowledge about the distribution of $D_{\mathcal{P}}^{\mathsf{intra}}$ is of great importance for characterizing the *reproducibility* of a PUF class (cf. Sect. 3.2.2). Therefore, estimated descriptive statistics of this distribution are commonly presented as one of the most important basic quality metrics for a PUF class. Based on the observed responses $\mathbf{Y}_{\mathsf{Exp}(\mathcal{P})}$ of a PUF experiment $\mathsf{Experiment}_{\mathcal{P}}(N_{\mathsf{puf}}, N_{\mathsf{chal}}, N_{\mathsf{meas}})$, a new array $\mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{intra}}$ of observed response intra-distances can be calculated:

$$\mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{intra}} = \left[\mathbf{dist}\left[y_i^{(j_1)}(x_k);\, y_i^{(j_2)}(x_k)\right]\right]_{\forall 1 \le i \le N_{\mathsf{puf}}; \forall 1 \le k \le N_{\mathsf{chal}}; \forall 1 \le j_1 \ne j_2 \le N_{\mathsf{meas}}}.$$

From this array of observed response intra-distances, the following descriptive statistics are often calculated to provide an estimation of the underlying distribution parameters of $D_{\mathcal{P}}^{\mathsf{intra}}$:

- An estimate of the distribution mean of $D_{\mathcal{P}}^{\mathsf{intra}}$ or $\mathsf{E}[D_{\mathcal{P}}^{\mathsf{intra}}]$ is obtained from the sample mean of $\mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{intra}}$:

$$\mu_{\mathcal{P}}^{\mathsf{intra}} = \overline{\mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{intra}}} = \frac{2}{N_{\mathsf{puf}} \cdot N_{\mathsf{chal}} \cdot N_{\mathsf{meas}} \cdot (N_{\mathsf{meas}} - 1)} \cdot \sum \mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{intra}}.$$

- Equivalently, an estimate of the standard deviation of the distribution of $D_{\mathcal{P}}^{\mathsf{intra}}$ or $\mathsf{\Sigma}[D_{\mathcal{P}}^{\mathsf{intra}}]$ is obtained as:

$$\sigma_{\mathcal{P}}^{\mathsf{intra}} = \sqrt{\frac{2}{N_{\mathsf{puf}} \cdot N_{\mathsf{chal}} \cdot N_{\mathsf{meas}} \cdot (N_{\mathsf{meas}} - 1) - 2} \cdot \sum \left(\mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{intra}} - \mu_{\mathcal{P}}^{\mathsf{intra}}\right)^2}.$$

- A estimate of the shape of the underlying distribution of $D_{\mathcal{P}}^{\mathsf{intra}}$ is given by the histogram of $\mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{intra}}$.
- Order statistics of $\mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{intra}}$ give more robust information about the distribution of $D_{\mathcal{P}}^{\mathsf{intra}}$ when it is skewed. Regularly used order statistics are the minimum, the maximum and the median, and for more detail the $1/4$- and $3/4$-quantiles and the 1 %- and 99 %-percentiles. In particular the maximum and the 99 %-percentile of $\mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{intra}}$ are of interest since they provide a good estimate of the largest PUF response intra-distances, i.e. the 'least reliable' PUF response one can expect.

Intra-distance Statistics Under Variable Evaluation Conditions

Variations in evaluation conditions such as environment temperature and supply voltage generally influence the intra-distance between PUF responses. The distance between two PUF responses evaluated on the same PUF instance and for the same challenge, but under different conditions α_1 and α_2, is typically larger than the intra-distance between the same responses evaluated under one fixed condition. For PUF-based applications, it is important to learn the worst-case, i.e. the largest, intra-distance which can arise for a given range of evaluation conditions with respect to a particular reference condition α_{ref}.

Definition 3 The PUF response intra-distance under condition α with respect to a reference condition α_{ref} is a random variable defined as:

$$D^{\mathsf{intra}}_{\mathsf{puf}_i;\alpha}(x) \stackrel{\triangle}{=} \mathbf{dist}\big[Y_i^{\alpha_{\mathsf{ref}}}(x);\, Y_i^{\alpha}(x)\big],$$

with $Y_i^{\alpha_{\mathsf{ref}}}(x) \leftarrow \mathsf{puf}_i(x).\mathsf{Eval}^{\alpha_{\mathsf{ref}}}$ and $Y_i^{\alpha}(x) \leftarrow \mathsf{puf}_i(x).\mathsf{Eval}^{\alpha}$ two distinct evaluations of PUF instance puf_i on the same challenge x but under different conditions α_{ref} and α. The PUF response intra-distance for a random PUF instance and a random challenge is defined as the random variable:

$$D^{\mathsf{intra}}_{\mathcal{P};\alpha} \stackrel{\triangle}{=} D^{\mathsf{intra}}_{\mathsf{PUF}\leftarrow\mathcal{P};\alpha}(X \leftarrow \mathcal{X}_{\mathcal{P}}).$$

All earlier introduced intra-distance statistics can be extended in the same manner to obtain estimates of the distribution of $D^{\mathsf{intra}}_{\mathcal{P};\alpha}$. In practice, the nominal operating condition is selected as the reference condition. To find the worst-case intra-distance in a given range of conditions, one typically evaluates the intra-distance under the extrema of this range. This makes sense under the reasonable assumption that intra-distance with respect to the reference condition increases if one moves further away from the reference. The extrema of a range of conditions are called the *corner cases*. As an example, assume one needs to find the worst-case intra-distance behavior over the environment temperature range $T_{env} = -40\ ^{\circ}\mathrm{C} \ldots 85\ ^{\circ}\mathrm{C}$. The reference condition is set at room temperature, $\alpha_{\mathsf{ref}} = (T_{env} = 25\ ^{\circ}\mathrm{C})$, and the intra-distances at the corner cases $\alpha_1 = (T_{env} = -40\ ^{\circ}\mathrm{C})$ and $\alpha_2 = (T_{env} = 85\ ^{\circ}\mathrm{C})$ are studied.

2.2.4 PUF Response Inter-distance

Inter-distance Definition

Definition 4 A PUF response **inter**-distance is a random variable describing the distance between two PUF responses from different PUF instances using the same challenge:

$$D^{\mathsf{inter}}_{\mathcal{P}}(x) \stackrel{\triangle}{=} \mathbf{dist}\big[Y(x);\, Y'(x)\big],$$

with $Y(x)$ and $Y'(x)$ evaluations of the same challenge x on two random but distinct PUF instances $\mathsf{PUF} \leftarrow \mathcal{P}$ and $\mathsf{PUF}'(\neq \mathsf{PUF}) \leftarrow \mathcal{P}$. Additionally, the PUF response inter-distance for a random challenge is defined as the random variable:

$$D_{\mathcal{P}}^{\mathsf{inter}} \triangleq D_{\mathcal{P}}^{\mathsf{inter}}(X \leftarrow \mathcal{X}_{\mathcal{P}}).$$

Inter-distance Statistics

Again, the distribution of the random variable $D_{\mathcal{P}}^{\mathsf{inter}}$ is an important metric for a PUF class, in this case to characterize the *uniqueness* of its instances. Based on the observed responses $\mathbf{Y}_{\mathsf{Exp}(\mathcal{P})}$ of a PUF experiment $\mathsf{Experiment}_{\mathcal{P}}(N_{\mathsf{puf}}, N_{\mathsf{chal}}, N_{\mathsf{meas}})$, a new array $\mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{inter}}$ of observed response inter-distances can be calculated:

$$\mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{inter}} = \left[\mathbf{dist}\left[y_{i_1}^{(j)}(x_k);\, y_{i_2}^{(j)}(x_k)\right]\right]_{\forall 1 \le i_1 \ne i_2 \le N_{\mathsf{puf}}; \forall 1 \le k \le N_{\mathsf{chal}}; \forall 1 \le j \le N_{\mathsf{meas}}}.$$

From this sample array of response inter-distances, the following descriptive statistics can be calculated to provide an estimation of the underlying distribution parameters of $D_{\mathcal{P}}^{\mathsf{inter}}$:

- The sample mean of $\mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{inter}}$ as an estimate of $\mathsf{E}[D_{\mathcal{P}}^{\mathsf{inter}}]$:

$$\mu_{\mathcal{P}}^{\mathsf{inter}} = \overline{\mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{inter}}} = \frac{2}{N_{\mathsf{puf}} \cdot (N_{\mathsf{puf}} - 1) \cdot N_{\mathsf{chal}} \cdot N_{\mathsf{meas}}} \cdot \sum \mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{inter}}.$$

- Equivalently, an estimate of $\Sigma[D_{\mathcal{P}}^{\mathsf{inter}}]$ is obtained as:

$$\sigma_{\mathcal{P}}^{\mathsf{inter}} = \sqrt{\frac{2}{N_{\mathsf{puf}} \cdot (N_{\mathsf{puf}} - 1) \cdot N_{\mathsf{chal}} \cdot N_{\mathsf{meas}} - 2} \cdot \sum \left(\mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{inter}} - \mu_{\mathcal{P}}^{\mathsf{inter}}\right)^2}.$$

- A estimate of the shape of the underlying distribution of $D_{\mathcal{P}}^{\mathsf{inter}}$ is given by the histogram of $\mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{inter}}$.
- The order statistics of $\mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{inter}}$ again give more robust information about the distribution of $D_{\mathcal{P}}^{\mathsf{inter}}$ when it is skewed. For PUF response inter-distances, in particular the minimum and the 1 %-percentile of $\mathbf{D}_{\mathsf{Exp}(\mathcal{P})}^{\mathsf{inter}}$ are of interest since they are good estimates of the smallest inter-distance, i.e. the 'least unique' PUF instance pair, one can expect.

Inter-distance Statistics Under Variable Evaluation Conditions

The inter-distance between PUF responses is also susceptible to variations in evaluation conditions. However, for typical PUF-based applications, the initial generation or enrollment is done in a secure environment under controlled conditions, and only

the inter-distance of evaluations under the reference condition α_{ref} is of importance. Therefore, no extension of the definition of inter-distance to varying conditions is required in that case. In applications where the initial enrollment is done in the field, the susceptibility of the inter-distance to the evalution conditions does need to be taken into account. In this book, we only consider enrollment at reference conditions.

2.3 Terminology and Classification

In Sect. 2.3.1, we elaborate on the origin and the meaning of the acronym 'PUF'. As will become clear from this chapter, the growing popularity of PUFs has caused a proliferation of different constructions and concepts all labelled as PUFs. In an attempt to bring some order into this zoo, a number of classification attempts were made:

- Based on the implementation technology of the proposed construction: non-electronic PUF versus electronic PUFs versus silicon PUFs, as discussed in Sect. 2.3.2.
- Based on a more general set of physical construction properties: non-intrinsic versus intrinsic PUFs, as discussed in Sect. 2.3.3.
- Based on the algorithmic properties of their challenge-response behavior: weak versus strong PUFs, as discussed in Sect. 2.3.4.

2.3.1 "PUFs: Physical(ly) Unclon(e)able Functions"

To PUF or Not to PUF?

In addition to proposing a new optics-based PUF construction, Pappu [104] in 2001 was the first to also describe and define the more general concept of a PUF, which he introduced at the time as a *physical one-way function*. Shortly after, Gassend et al. [42] proposed a new silicon-based PUF construction and defined it similarly as a *physical random function*, but they opted for the acronym *PUF*, standing for *physical unclonable function*, to avoid confusion with the concept of a pseudo-random function which in cryptography is already abbreviated as PRF.

In the meantime, the acronym 'PUF' has become a catch-all for a variety of often very different constructions and concepts, but which share a number of interesting properties. Some of these constructions were proposed before the term PUF was coined, and were hence not called PUFs from the beginning. Other constructions were proposed in fields other than cryptographic hardware, where the term PUF is yet unknown. Depending on which properties are assumed to be defining for PUFs (and depending on whom you ask), such constructions are still called PUFs.

PUF having been used as a label for many different constructions, the literal semantic meaning of the acronym as "physical unclonable function" has been partially lost, up to the point where some consider it a misnomer. Moreover, slight variations in the actual wording were introduced over time, as expressed in the title of this section. In this book, the acronym PUF does not generally refer to its literal meaning. Instead, it is used as the collective noun for the variety of proposed constructions sharing a number of interesting properties. One of the main goals of the next chapter is to list known PUF constructions and identify a least common subset of such defining properties.

Physical Versus Physically?

Originally, the acronym PUF stood for *physical* unclonable function, but later the variant *physically* unclonable function also got into use. Both terms are still used today and refer to the same concept. The choice for one or the other depends mainly on the preference of the author. However, from a strictly semantic point of view, there is a small difference in meaning. A *physical* unclonable function is 'a physical function which is unclonable' whereas a *physically* unclonable function is 'a function which is physically unclonable'. In Sect. 3.2.10, we argue why we consider the second interpretation to be a more fitting description of the actual concept of a PUF.

Unclon(e)able?

The adjective *unclonable* is the central specifier in the acronym, reflecting the distinguishing property of a PUF. However, the actual meaning of unclonable is not clearly defined. In a broader sense, one first needs to define what one considers a *clone*. Especially in the context of PUFs, this has led to some debate. In many scenarios, one solely observes the input-output (or challenge-response) behavior of a PUF and only rarely the actual physical entity. In that respect, any construction which accurately mimics the behavior of a particular PUF could be considered a clone thereof. On the other hand, if one takes the physical aspect into account, cloning gets a different meaning, e.g. one could require a clone to have the same behavior *and* the same physical appearance as the original, or even more strictly, one could require a clone to be manufactured in the same production process as the original. The distinction between the former and latter interpretation of a clone is often made specific as a *mathematical* versus a *physical* clone.

Whether or not to write the silent '*e*' in unclon(e)able is somewhat arbitrary. Generally, the terminal, silent *-e* is dropped when appending the suffix *-able* to a verb [35], e.g. *movable*, *blamable*, unless the verb ends in *-ce* or *-ge*, which is not the case here. However, in scientific literature on computing the form (un)clon*e*able is more common, likely due to the existence of an homonymous but unrelated interface in the popular Java programming language [58]. A web search gives roughly a five-to-one preference of *unclonable* over *uncloneable*, although peculiarly for *clonable* versus *cloneable* these odds are reversed. In this book, the form *unclonable* is preferred.

Function?

In the pure mathematical sense, a PUF is not even strictly a *function*, since a single input (challenge) can be related to more than a single output (response) due to the uncontrollable effects of the physical environment and random noise on the response generation. Therefore, a more fitting mathematical description of a PUF is as a *probabilistic function*, i.e. a function for which part of the input is an uncontrollable random variable. This is sometimes made explicit by providing an additional input r to a function, representing an (undetermined) number of fair coin tosses: $r \xleftarrow{\$} \{0,1\}^*$. Alternatively, one can consider the outcome of a probabilistic function to be a random variable with a particular probability distribution, instead of a deterministic value.

2.3.2 Non-electronic, Electronic and Silicon PUFs

PUFs and PUF-like constructions have been proposed based on a wide variety of technologies and materials such as glass, plastic, paper, electronic components and silicon integrated circuits (ICs). A possible classification of PUF constructions is based on the *electronic* nature of their identifying features.

Non-electronic PUFs

A first, relatively large class contains PUF proposals with an inherently non-electronic nature. Their PUF-like behavior is based on non-electronic technologies or materials, e.g. the random fiberstructure of a sheet of paper or the random reflection of scattering characteristics of an optical medium (cf. Sect. B.1). Note that the denomination non-electronic reflects the origin of the PUF behavior, and not the way PUF responses are processed or stored, which often does use electronics and digital circuitry.

Electronic PUFs

On the other side of the spectrum are electronic PUFs, i.e. constructions containing random variations in their electronic characteristics, e.g. resistance, capacitance, etc. These variations are generally also measured in an electronic way. Note that there is not always a sharp distinction between electronic and non-electronic PUFs, e.g. a construction called RF-DNA [29] consists of randomly arranged copper wires fixated in a flexible silicone medium which could be considered non-electronic, whereas LC-PUFs [46] are in fact passive LC resonance circuits which could be considered electronic in nature. However, the manner in which they generate responses is practically the same, i.e. the random influence they have on an

imposed wireless radio-frequency field (cf. Sect. B.2). Another side note is that variation in electronic behavior is practically always a consequence of variations in non-electronic parameters at a lower level of physical abstraction, e.g. the length and section of a metal wire determine its resistance, the area and distance between two conductors determine their capacitance, etc.

Silicon PUFs

A major subclass of electronic PUFs are the so-called silicon PUFs, i.e. integrated electronic circuits exhibiting PUF behavior which are embedded on a silicon chip. The term silicon PUF and the first practical realization of a PUF structure in silicon were introduced by Gassend et al. [42]. Since silicon PUFs can be directly connected to standard digital circuitry embedded on the same chip, they can be immediately deployed as a hardware building block in cryptographic implementations. For this reason, it is evident that silicon PUFs are of particular interest for security solutions and they are the main type of PUF construction we focus on in this book.

2.3.3 Intrinsic and Non-intrinsic PUFs

An important classification of PUFs based on their construction properties is that of *intrinsic PUFs* as initially proposed by Guajardo et al. [45]. In a slightly adapted form, we consider that a PUF construction needs to meet at least two conditions to be called an intrinsic PUF: (i) its evaluations are performed *internally* by embedded measurement equipment, and (ii) its random instance-specific features are *implicitly* introduced during its production process. Next, we discuss both properties in more detail and highlight the specific practical and security advantages of intrinsic PUFs.

External Versus Internal Evaluations

A PUF evaluation is a physical measurement which can take many forms. We distinguish between external and internal evaluations. An *external* evaluation is a measurement of features which are externally observable and/or which is carried out using equipment which is external to the physical entity containing the features. On the other side are *internal* evaluations, i.e. measurements of internal features which are carried out by equipment completely embedded in the instance itself. This might seem a rather arbitrary distinction, but there are some important advantages to internal evaluations:

- The first is a purely practical advantage. Having the measurement equipment embedded in every PUF instance means that every PUF instance can evaluate itself without any external limitations or restrictions. Internal evaluations are typically also more accurate since there is less opportunity for outside influences and measurement errors.

- Secondly, there is a rather important security advantage to internal evaluations, since the PUF responses originate inside the embedding object. This means that, as long as an instance does not disclose a response to the outside world, it can be considered an internal secret. This is of particular interest when the embedding object is a digital IC, since in that case the PUF response can be used immediately as a secret input for an embedded implementation of a cryptographic algorithm.

A possible disadvantage of an internal evaluation is that one needs to trust the embedded measurement equipment, since it is impossible to externally verify whether the measurement takes place as expected, this as opposed to an external evaluation, for which it is often possible to actually observe the measurement in progress.

Explicit Versus Implicit Random Variations

A second construction-based distinction considers the source of the randomness of the evaluated features in a PUF. The manufacturing process of the PUF instances can contain an *explicit* randomization procedure with the sole purpose of introducing random features which will later be measured when the PUF is evaluated. Alternatively, the measured random features can also be an *implicit* side effect of the standard production flow of the embedding object, i.e. they arise naturally as an artifact of uncontrollable effects during the manufacturing process. Again, there are some subtle but interesting advantages to PUF constructions based on implicit random variations:

- From a practical viewpoint, implicit random variations generally come at no extra cost since they are already (unavoidably) present in the production process. Introducing an explicit randomization procedure on the other hand can be costly, particularly due to timing constraints in the manufacturing flow of high-volume products.
- Implicit random variability in manufacturing or so-called *process variations* which occur during a production process are generally undesirable since they have a negative impact on yield. This means that manufacturers already take elaborate measures to reduce process variations as much as possible. However, as it turns out, completely avoiding any random effect during manufacturing is technically impossible and process variations are always present. As a consequence, PUF constructions based on implicit process variations have the interesting security advantage that even the manufacturers, having all details and full control over their production process, cannot remove or control the random features at the core of the PUF's functionality.

Non-intrinsic PUF Constructions

A variety of PUF constructions have been proposed which we label *non-intrinsic* according to this classification, e.g.:

- The optical PUF as proposed by Pappu et al. [104, 105], which is based on the unique speckle pattern which arises when shining a laser on a transparent material with randomly solved scattering particles.
- The coating PUF as proposed by Tuyls et al. [147], which is based on the unique capacitance between two metal lines in a silicon chip whose top layer consists of a coating with randomly distributed dielectric particles.

The optical PUF is non-intrinsic since it is externally evaluated (by observing the speckle pattern) and its random features are explicitly introduced (by solving the scattering particles). The coating PUF is also non-intrinsic: although it can be internally evaluated on a silicon chip, its randomness is still explicitly generated during production (by solving the dielectric particles). Since our main focus is on intrinsic PUFs, we will not go into more detail on the construction of non-intrinsic PUFs in the main text of this book. For completeness, we have added a detailed description of many known non-intrinsic PUF constructions in Appendix B.

2.3.4 Weak and Strong PUFs

A last classification of PUFs we discuss here is explicitly based on the security properties of their challenge-response behavior. The distinction between *strong* and *weak* PUFs was first introduced by Guajardo et al. [45], and further refined by Rührmair et al. [116]. Basically, a PUF is called a strong PUF if, even after giving an adversary access to a PUF instance for a prolonged period of time, it is still possible to come up with a challenge to which with high probability the adversary does not know the response. Note that this implies at least that: (i) the considered PUF has a very large challenge set, since otherwise the adversary can simply query all challenges and no unknown challenges are left, and (ii) it is infeasible to built an accurate model of the PUF based on observed challenge-response pairs, or in other words the PUF is unpredictable. PUFs which do not meet the requirements for a strong PUF, and in particular PUFs with a small challenge set, are consequentially called weak PUFs. An extreme case is a PUF construction which has only a single challenge. Such a construction is also called a physically obfuscated key or POK (cf. Sect. 2.5.1). It is evident that the security notion of a strong PUF is higher than that of a weak PUF, hence enabling more applications. However constructing a practical (intrinsic) strong PUF with strong security guarantees turns out to be very difficult, up to the point where one can consider it an open problem whether this is actually possible.

2.4 Intrinsic PUF Constructions

All known intrinsic PUF constructions are silicon PUFs based on random process variations occurring during the manufacturing process of silicon chips. The known intrinsic silicon PUF constructions can be further classified based on their operating principles:

- A first class of constructions measures random variations on the delay of a digital circuit; the constructions are therefore called *delay-based* silicon PUFs.
- A second class of silicon PUF constructions use random parameter variations between matched silicon devices, also called device mismatch, in bistable memory elements. These constructions are labelled *memory-based* silicon PUFs.
- Finally, we classify a collection of intrinsic silicon PUF proposals consisting of mixed-signal circuits. Their basic operation is of an analog nature, which means an embedded analog measurement is quantized with an analog-to-digital conversion to produce a digital response representation.

2.4.1 Arbiter PUF

Basic Operation and Construction

The arbiter PUF, as proposed by Lee et al. [43, 75, 78], is a type of delay-based silicon PUF. The idea behind an arbiter PUF is to explicitly introduce a race condition between two digital paths on a silicon chip. Both paths end in an *arbiter circuit* which is able to resolve the race, i.e. it determines which of the two paths was faster and outputs a binary value accordingly. If both paths are designed or configured to have (nearly) identical nominal delays, the outcome of the race, and hence the output of the arbiter, cannot be unambiguously determined based on the design. Two scenarios are possible:

1. Even when designed or configured with identical nominal delays, the actual delay experienced by an edge which travels the two paths will not be exactly equal. Instead, there will be a random delay difference between the two paths, due to the effect of random silicon process variations on the delay parameters. This random difference will determine the race outcome and the arbiter output. Since the effect of silicon process variations is random per device, but static for a given device, the delay difference and by consequence the arbiter output will be device-specific. This is the basis for the PUF behavior of an Arbiter PUF.
2. Due to their stochastic nature, there is a non-negligible possibility that by chance both delays are nearly identical. In that case, when two edges are applied simultaneously on both paths they will reach the arbiter virtually at the same moment. This causes the arbiter circuit to go into a *metastable* state, i.e. the logic output of the arbiter circuit is temporarily undetermined (from an electronic point of view, the voltage of the arbiter output is somewhere in between the levels for a logic high and low). After a short but random time, the arbiter leaves its metastable state and outputs a random binary value which is independent of the outcome of the race. Although random, in this case the arbiter's output is not device-specific and not static. This phenomenon is the cause of unreliability (noise) of the responses of an arbiter PUF.

Lee et al. [43, 75, 78] propose implementing the two delay paths as chains of *switch blocks*. A switch block connects two input signals to its two outputs in one of

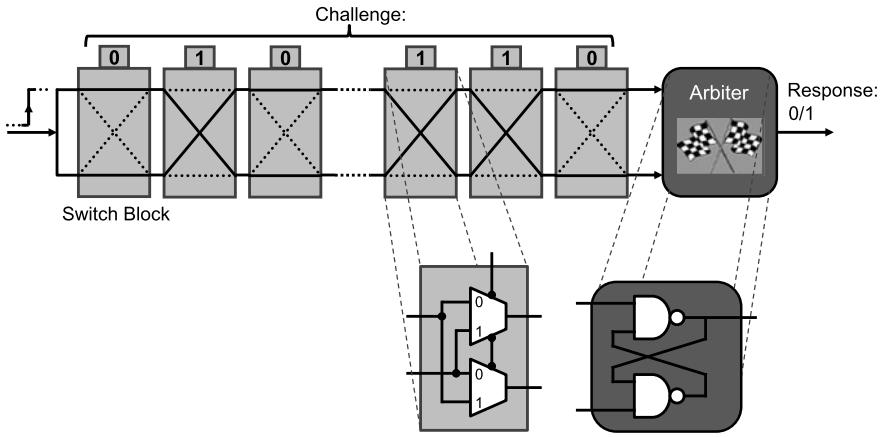


Fig. 2.1 Construction of a basic arbiter PUF as proposed by Lee et al. [43, 75, 78]

two possible configurations, straight or crossed, based on a configurable selection bit. The logic functionality of a switch block can be implemented in a number of ways, the most straightforward being with a pair of 2-to-1 multiplexers (muxes). By concatenating n switch blocks, a total of n configuration bits are needed to configure any of 2^n possible pairs of delay paths. This n-bit setting is considered the challenge of the arbiter PUF. Since the input-output delays of the two configurations of a switch block are slightly different and randomly affected by process variations, each of these 2^n challenges leads to a new race condition which can be resolved by the arbiter. The resulting output bit of the arbiter is considered the response of the arbiter PUF to the given challenge. However, it is immediately clear that these 2^n different configurations are not independent, since they are only based on a number of underlying delay parameters linear in n. This will lead to challenge-response modeling attacks on arbiter PUFs as we will discuss later. The arbiter circuit can also be implemented in a number of ways, the most straightforward being as an SR NAND latch. Most types of digital latches and flip-flops exhibit arbiter behavior. The construction of the basic arbiter PUF is depicted in Fig. 2.1

Ideally, an arbiter PUF response is solely determined by the random effect of process variations on the delay parameters of the circuit. This can only happen when two conditions are met:

1. The delay lines are designed to be nominally perfectly symmetrical, i.e. any difference in delay is solely caused by process variations.
2. The arbiter circuit is completely fair, i.e. it does not favor one of its inputs over the other. Lin et al. [80] conclude that a basic SR latch is the best option since it is unbiased due to its symmetric construction.

If either of the two conditions is not met, the arbiter PUF is biased, which results in a lower uniqueness of its responses. Designing a perfectly unbiased arbiter PUF is highly non-trivial since it requires very low-level control over implementation choices. In some technologies, e.g. on FPGAs [100], this is not possible, which

greatly impedes the successful deployment of arbiter PUFs on them. If bias is unavoidable, some unbiasing techniques are possible. A tunable delay circuit in front of the arbiter PUF can counteract the deterministic bias of the delay paths or the arbiter, e.g. if the arbiter favors the upper delay path then the tunable delay can give the lower path a head start. A relatively simple way of doing this is by fixing a number of challenge bits in the beginning of the switch block chain in such a way as to counteract the bias.

Discussion on Modeling Attacks

The basic arbiter PUF construction with n chained switch blocks provides us with 2^n different challenges. However, the number of delay parameters which determine the arbiter PUF response is only linear in n, which means that these 2^n challenges cannot produce independent responses. In practice, when one learns the underlying delay parameters and one can model the interaction with the challenge bits, one is able to accurately predict response bits to random challenges, even without access to the PUF. In many ways, such a model can be considered a clone of the arbiter PUF, be it a mathematical clone. Arbiter PUF responses are only considered unpredictable as long as such a model cannot be accurately built.

It was immediately realized by Lee et al. [75] that their basic arbiter PUF can indeed be easily modeled and they even present modeling attack results for their own implementation. This and later modeling attacks assume a simple but accurate additive delay model for the arbiter PUF, i.e. the delay of a digital circuit is modeled as the sum of the delays of its serialized components. This turns out to be a very accurate first-order approximation for the relatively simple switch block chains. The unknown underlying delay parameters are learned from observing challenge-response pairs of the PUF. Every challenge-response pair constitutes a linear inequality over these parameters and is used to obtain a more accurate approximation. Moreover, the unknown parameters do not need to be calculated explicitly, but they are learned implicitly and automatically by an appropriate *machine-learning technique*. Machine-learning techniques such as artificial neural networks (ANNs) and support-vector machines (SVMs) are able to generalize behavior based on applied examples. Numerous results have been presented which successfully apply different machine-learning techniques in order to model the behavior of an arbiter PUF. The ultimate goal of such modeling attacks is to predict unknown responses as accurately as possible after having been trained with as few examples as possible. We say that a PUF is $(p_{\mathsf{model}}, q_{\mathsf{train}})$-modelable if a known modeling attack exists which is able to predict unseen responses of the PUF with a success rate p_{model}, after training the model with q_{train} known challenge-response pairs.

Experimental Results and Extended Constructions

Arbiter PUFs were initially implemented by Gassend et al. [43] on a set of 23 FPGA devices. From experimental results, the average intra- and inter-distance of

the responses were estimated as $\mu_{\mathcal{P}}^{\text{intra}} = 0.1$ % and $\mu_{\mathcal{P}}^{\text{inter}} = 1.05$ %. The inter-distance is particularly low, with on average only one in 100 challenges producing a response bit which actually differs among different PUF instances. This arbiter PUF implementation is hence very biased, which is a result of the lack of low-level control over the placement and routing of the delay lines on the FPGA platform. A following implementation by the same team on a set of 37 ASICs proves to be less biased with $\mu_{\mathcal{P}}^{\text{inter}} = 23$ %, but is still far from the ideal of 50 %. The reliability of this implementation is very high, with $\mu_{\mathcal{P}}^{\text{intra}} = 0.7$ % under reference conditions. This worsens to $\mu_{\mathcal{P}}^{\text{intra}} = 4.82$ % when temperature rises to $T_{env} = 67$ °C. Lim [78] demonstrates that this arbiter PUF implementation is $(p_{\text{model}} = 96.45\ \%, q_{\text{train}} = 5000)$-modelable by applying an SVM-based machine-learning modeling attack. This result is problematic for this implementation, since the expected prediction error of $100\ \% - 96.45\ \% = 3.55\ \%$ is smaller than the average unreliability of the PUF under temperature variation. In practice, this means this implementation is not secure (after observing $q_{\text{train}} = 5000$ challenge-response pairs).

All subsequent work on arbiter PUFs is an attempt to make model-building attacks more difficult, by introducing non-linearities in the delays and by controlling and/or restricting the inputs and outputs to the PUF. Lee et al. [75] propose *feed-forward* arbiter PUFs as a first attempt to introduce non-linearities in the delay lines. They are an extension of their basic arbiter PUF, where the configuration of some switch blocks in the delay chains is not externally set by challenge bits, but is determined by the outcome of intermediate arbiters evaluating the race at intermediate points in the delay lines. This was equivalently tested on 37 ASICs, leading to $\mu_{\mathcal{P}}^{\text{inter}} = 38$ % and $\mu_{\mathcal{P}}^{\text{intra}} = 9.84$ % (at $T_{env} = 67$ °C). Note that the responses are much noisier, which is caused by the increased metastability since there are multiple arbiters involved. It was shown that the simple model-building attacks which succeeded in predicting the simple arbiter don't work any longer for this non-linear arbiter PUF. However, later results by Majzoobi et al. [93] and Rührmair et al. [118] show that with more advanced modeling techniques it is still possible to build an accurate model for the feed-forward arbiter PUF. Based on simulated implementations of feed-forward arbiter PUFs, Rührmair et al. [118] demonstrate them to be $(p_{\text{model}} > 97.5\ \%, q_{\text{train}} = 50000)$-modelable, with the actual success rate depending on the delay path length and the number of feed-forward arbiters. More advanced feed-forward architectures are proposed by Lao and Parhi [74].

Majzoobi et al. [94] present an elaborate discussion on techniques to make arbiter-based PUFs on FPGA more resistant to modeling. They use an initial device characterization step to choose the optimal parameters for a particular instantiation and use the reconfiguration possibilities of FPGAs to implement this. To increase randomness and to thwart model-building attacks, they use "hard-to-invert" input and output networks controlling the inputs to and outputs of the PUF, although these are not shown to be cryptographically secure. In particular, they consider multiple arbiter PUFs in parallel and take the exclusive-or (XOR) of the arbiter evaluations as the new response. By simulation, they show that this construction gives desirable PUF properties and makes model-building much harder. However,

Rührmair et al. [118] again show that model-building of these elaborate structures might be feasible by presenting a ($p_{\text{model}} = 99\ \%, q_{\text{train}} = 6000$)-model building attack based on a simulated implementation consisting of $n = 64$ switch blocks and three parallel arbiters which are XOR-ed together. For longer delay paths and more parallel arbiters, this modeling attack keeps on working but requires a considerable larger q_{train}.

Finally, a different approach towards model-building attacks for arbiter PUFs is taken by Öztürk et al. [48, 102, 103]. Instead of preventing the attack, they use the fact that a model of the PUF can be constructed as part of an authentication protocol. The proposed protocol is a variant of the protocol by Hopper and Blum [54], with security based on the difficulty of learning parity with noise. This is a highly unconventional use of a PUF as it explicitly assumes the PUF *can* be cloned mathematically, with the cloned PUF model acting as a noisy shared secret.

2.4.2 Ring Oscillator PUF

Basic Operation and Construction

A second type of delay-based intrinsic PUF is the ring oscillator PUF. A number of different constructions can be placed under this type, all based on measuring random variations on the frequencies of digital oscillating circuits. The source of this randomness is again the uncontrollable effect of silicon process variations on the delay of digital components. When these components constitute a ring oscillator, the frequency of this oscillation is equivalently affected. A typical ring oscillator PUF construction has two basic components, ring oscillators and frequency counters, which are arranged in a particular architecture and are often combined with a response-generating algorithm.

The first type of ring oscillator PUF was proposed by Gassend et al. [40, 42]. The ring oscillator they use is a variant of the switch block-based delay line as proposed for the arbiter PUF (cf. Sect. 2.4.1). This delay circuit is transformed into an oscillator by applying negative feedback. An additional AND-gate in the loop allows us to enable/disable the oscillation. The oscillating signal is fed to a frequency counter which counts the number of oscillating cycles in a fixed time interval. The resulting counter value is a direct measure of the loop's frequency. The construction of such a very basic ring oscillator PUF is shown in Fig. 2.2. Gassend et al. [40, 42] propose using a synchronously clocked counter. The oscillating signal is first processed by a simple edge detector which enables the counter every time a rising edge is detected. This architecture is robust, but the use of an edge detector limits the frequency of the ring oscillator to half the clock frequency.

The measured frequency of equally implemented ring oscillators on distinct devices shows sufficient variation due to process variations to act as a PUF response. However, it was immediately observed that the influence of environmental conditions on the evaluation of this construction is significant, i.e. changes in temperature

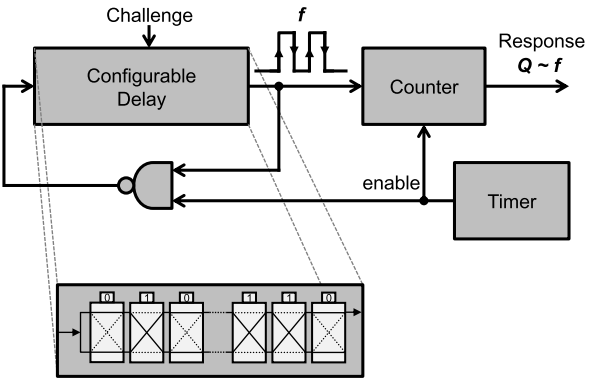


Fig. 2.2 Construction of a simple ring oscillator PUF as proposed by Gassend et al. [42]

and voltage cause frequency changes which are orders of magnitude larger than those caused by process variations. To counter this influence, Gassend et al. [40, 42] propose an important post-processing technique called *compensated measuring*. The principle behind compensated measuring is to evaluate the frequency of two ring oscillators on the same device simultaneously and consider a differential function, i.e. the ratio, of both measurements as the eventual response. The reasoning behind this is that environmental changes will affect both frequencies in roughly the same way, and their ratio will be much more stable. Note that this type of differential measurement was already implied in the arbiter PUF construction considering two delay paths in parallel, which explains its relatively high reliability. For the proposed ring oscillator PUF construction, the compensated measurements based on the ratio of two frequencies also proves to be particularly effective. The average observed intra-distance (Euclidean) between evaluations is roughly two orders of magnitude smaller than the average inter-distance, even under varying temperature and voltage conditions.

Variants and Experimental Results

The results from Gassend et al. [42] clearly show the potential of ring oscillator PUFs. However, their proposed construction has some minor drawbacks. Firstly, since the ring oscillator is based on the same delay circuit as the simple arbiter PUF, this construction will also be susceptible to modeling attacks and similar countermeasures need to be included. Moreover, the produced response is an integer counter value, or a real value in the case of compensated measurement, and cannot be used directly as a bit string in subsequent building blocks. It first needs to be quantized in an appropriate way to obtain a bit string response with desirable distribution properties. Both these additions are indispensable in order to use this construction in an efficient and secure manner, but require additional resources.

Suh and Devadas [136] propose an alternative ring oscillator PUF architecture that is not susceptible to these modeling attacks and that naturally produces bitwise responses. Instead of a challengeable ring oscillator based on the delay circuit of

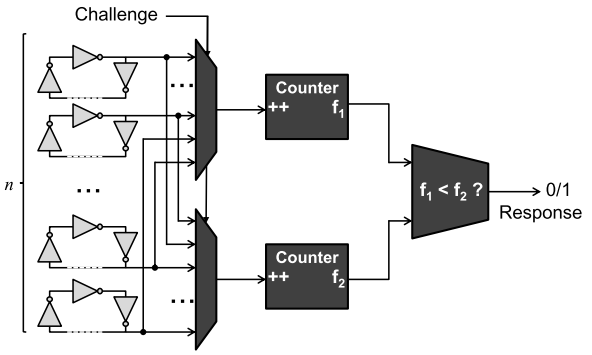


Fig. 2.3 Construction of a comparison-based ring oscillator PUF as proposed by Suh and Devadas [136]

an arbiter PUF, their architecture contains an array of n fixed but identically implemented inverter chains. The architecture also contains two frequency counters, which can be fed by each of the n inverter chains. Two n-to-1 muxes control which oscillators are currently applied to the two counters. The selection signals of these muxes constitute the PUF's challenge. The schematic construction of this PUF is depicted in Fig. 2.3. As in the earlier proposal, both frequency counters are enabled for a fixed time interval and their resulting counter values are measures of the frequencies of the two applied oscillators. The PUF's response bit is generated by comparing the two counter values. Since the exact frequencies of the inverter chains are influenced by process variations, the resulting comparison bit will be random and device-specific. Note that the comparison operation is a very basic form of a compensated measurement, i.e. an inverter chain which is faster than another one under certain conditions is likely to be faster under all conditions.

By considering many pairwise combinations of inverter chains, this ring oscillator PUF construction can produce many response bits. If the PUF implements n ring oscillators, a total of $\binom{n}{2} = \frac{n \cdot (n-1)}{2}$ pairs can be formed. However, it is clear that all these pairs do not produce independent evaluations, e.g. if oscillator A is faster than B and B is faster than C, then it is apparent that A will also be faster than C and the resulting response bit is dependent on the previous two. Suh and Devadas [136] correctly state that the number of independent comparisons one can make is limited by the number of possible ways the frequencies of the oscillators can be ordered. If the frequencies are independent and identically distributed, each of $n!$ possible orderings is equally likely and the maximal number of independent comparisons one can make is $\log_2 n!$. However, the list of independent comparisons is difficult to obtain and is moreover device-specific, i.e. different oscillator pairs need to be compared on each instance of the PUF. A simpler and device-generic approach is to compare fixed pairs and use every oscillator only once. In this way, a ring oscillator PUF consisting of n oscillators will only produce $\frac{n}{2}$ response bits which are however guaranteed to be independent. Suh and Devadas [136] reduce the number of response bits even further by applying a post-processing technique called *1-out-of-k masking*, i.e. they evaluate k oscillators and only consider the pair with the largest difference in frequency and output the result of this comparison. This technique

greatly enhances the reliability of the responses, but it comes at a relatively large resource overhead, i.e. out of n oscillators only $\lfloor \frac{n}{k} \rfloor$ response bits are generated. Experiments on a set of 15 FPGAs with $k = 8$ produce responses with an average inter-distance of 46.15 % and a very low average intra-distance of merely 0.48 % even under large temperature ($T_{env} = 120$ °C) and voltage ($V_{dd} + 10$ %) variations.

Maiti et al. [91] present results from a large-scale characterization of the ring oscillator PUF construction from Suh and Devadas [136] on FPGA. They do not apply the masking technique, but instead compare all neighbouring oscillators on the FPGA, yielding $n - 1$ response bits from n oscillators. Their results are particularly valuable as they are based on a large population of devices (125 FPGAs), which is rare in other works on PUFs. They present an average inter-distance of 47.31 % and an average intra-distance of 0.86 % at nominal conditions. However, changing temperature and especially voltage conditions have a significant impact on the intra-distance. At a 20 % reduced supply voltage, the average intra-distance goes up to 15 %. The measurement data resulting from this experiment is made publicly available [110].

Yin and Qu [157] build upon the construction from Suh and Devadas [136] and propose a number of adaptations which significantly improve the resource usage of the ring oscillator PUF. Firstly, they propose a generalization of the 1-out-of-k masking scheme which is much less aggressive in ignoring oscillators but achieves similar reliability results. The group of oscillators is partitioned into mutually exclusive subsets such that the minimal frequency difference between two members of a subset is above a certain threshold. An evaluation between a pair of oscillators from the same subset is therefore guaranteed to be very stable. Yet by using a clever grouping algorithm, practically all oscillators contribute to a response bit and only a few are ignored. Secondly, they propose a hybrid architecture for arranging ring oscillators and frequency counters. This allows us to find an optimal trade-off between the speed of generating responses and the resource usage of oscillators, counters and multiplexers.

Maiti and Schaumont [89, 90] also expand upon the construction from Suh and Devadas [136] and present some significant improvements. Firstly, they propose a number of guidelines for reducing systematic bias when implementing arrays of ring oscillators on an FPGA, and study the effect of each guideline on the uniqueness of the responses. Secondly, they propose a very efficient but highly effective variant of the 1-out-of-k masking scheme based on configurable ring oscillators. Instead of considering the most reliable pair out of k distinct oscillators, they make use of the oscillators' configurability and consider the most reliable out of k configurations of a single pair of oscillators. This achieves similar or even better reliability results than the original masking scheme, however without any sacrifice in resources.

A further improvement in the post-processing of ring oscillator PUFs is proposed by Maiti et al. [67, 92]. They go beyond ranking-based methods of oscillator frequencies by extracting a response based on the magnitude of the observed frequency differences. An elaborate post-processing algorithm based on test statistics of the observed frequency values and an identity mapping function is proposed. This substantially increases the amount of challenge-response pairs which can be

considered for a limited set of ring oscillators. Maiti et al. [92] acknowledge that responses to different challenges are no longer information-theoretically independent, but they still show strong PUF behavior in terms of their inter- and intra-distance distributions. Experimental data obtained from an implementation on 125 FPGAs with 65519 challenge evaluations demonstrates an average inter-distance of 49.99 % and an average intra-distance of 10 % at $T_{env} = 70$ °C. The proposed post-processing technique is computationally intensive, requiring a sophisticated datapath architecture.

2.4.3 Glitch PUF

A third type of delay-based PUF construction is based on glitch behavior of combinatorial logic circuits. A purely combinatorial circuit has no internal state, which means that its steady-state output is entirely determined by its input signals. However, when the logical value of the input changes, transitional effects can occur, i.e. it can take some time before the output assumes its steady-state value. These effects are called glitches and the occurrence of glitches is determined by the differences in delay of the different logical paths from the inputs to an output signal. Since the exact circuit delays of a particular instance of a combinatorial circuit are influenced by random process variations, the occurrence, the number and the shape of the glitches on its output signals will equivalently be partially random and instance-specific. When accurately measured, the glitch behavior of such a circuit can be used as a PUF response.

Anderson [1] proposes a glitch-based PUF construction specifically for FPGA platforms. A custom logical circuit is implemented which, depending on the delay variations in the circuit, does or does not produce a single glitch on its output. The output is connected to the preset signal of a flip-flop which captures the glitch, should it occur. This is the circuit's single response bit. Placing many of these (small) circuits on an FPGA allows us to produce many PUF response bits. A challenge selects a particular circuit. Experimental results of 36 PUF implementations each producing 128 response bits show an average inter-distance of 48.3 % and an average intra-distance of 3.6 % under high temperature conditions ($T_{env} = 70$ °C).

Shimizu et al. [129, 137] present a more elaborate glitch PUF construction and also introduce the term *glitch PUF*. They propose a methodology to use the glitch behavior of any combinatorial circuit as a PUF response and apply it specifically to a combinatorial FPGA implementation of the SubBytes operation of AES [28]. Their initial proposal [137] consists of an elaborate architecture which performs an on-chip high-frequency sampling of the glitch wave form and a quantization circuit which generates a response bit based on the sampled data. They propose using the parity of the number of detected glitches as a random yet robust feature of a glitch wave form. In a later version of their PUF construction [129], they propose a much simpler implementation which achieves basically the same functionality and results. By simply connecting a combinatorial output to a toggle flip-flop, the value of the

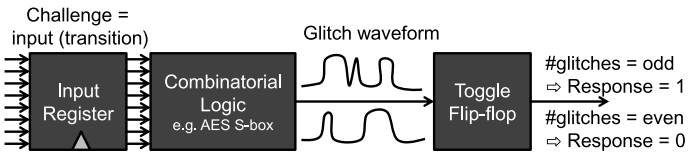


Fig. 2.4 Construction of a glitch PUF as proposed by Shimizu et al. [129]

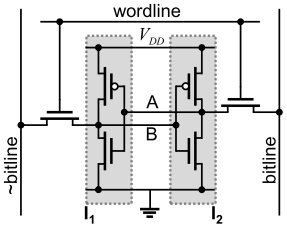
toggle flip-flop after the transitional phase will equal the parity of the number of glitches that occurred. This construction is shown in Fig. 2.4. To improve the reliability of the PUF responses, a *bit-masking* technique is used. During an initial measurement on every instance, unstable response bits are identified as responses that do not produce a stable value on m consecutive evaluations. In later measurements, these response bits are ignored, i.e. they are *masked*. The mask information needs to be stored alongside each PUF instance or in an external storage. This type of masking also causes a resource overhead, as the identified unstable bits are not used in the PUF responses. However, the resulting unmasked bits show a greatly improved reliability. Experimental results from 16 FPGA implementations of this glitch PUF construction on 2048 challenges show an average inter-distance of 35 % and an average intra-distance of 1.3 % at nominal conditions, when bit-masking is applied. The bit-masking results in about 38 % of the response bits being ignored. Under variations of the environmental conditions, the average intra-distance increases substantially, to about 15 % in the worst-case corner condition of $T_{env} = 85$ °C and $V_{dd} + 5$ %.

2.4.4 SRAM PUF

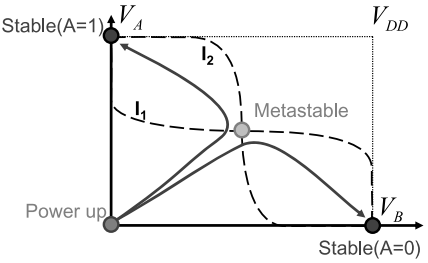
SRAM Background

Static Random-Access Memory or SRAM is a digital memory technology based on bistable circuits. In a typical CMOS implementation, an individual SRAM cell is built with six transistors (MOSFETs), as shown in Fig. 2.5a. The logic memory functionality of a cell comes from two cross-coupled inverters at its core, shown in Fig. 2.5c, each built from two MOSFETs, one p-MOS and one n-MOS. From an electronic viewpoint, this circuit contains a positive feedback loop which reinforces its current state. In a logic sense, this circuit has two stable values (bistable), and by residing in one of the two states the cell stores one binary digit. Two additional access MOSFETs are used to read and write its contents. Typically, many SRAM cells are arranged in large memory array structures, capable of storing many kilobits or megabits. An SRAM cell is volatile, meaning that its state is lost shortly after power-down.

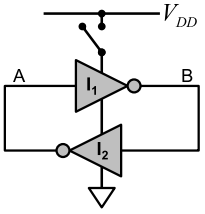
The detailed operation of an SRAM cell is best explained by drawing the voltage transfer curves of the two cross-coupled inverters, shown in Fig. 2.5b. From this graph, it is clear that the cross-coupled CMOS inverter structure has three possible operating points of which only two are stable and one is metastable. The sta-



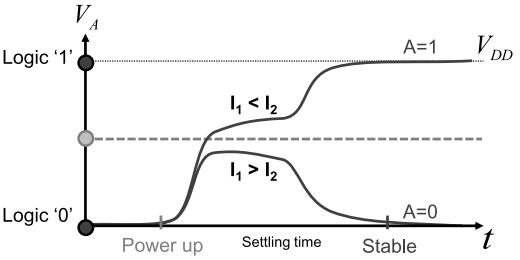
(a) SRAM cell CMOS circuit



(b) SRAM cell voltage transfer curves



(c) SRAM cell logic circuit



(d) SRAM cell power-up transient analysis

Fig. 2.5 Construction and power-up behavior of an SRAM cell

ble points are characterized by the property that deviations from these points are reduced and the stable point condition is restored. This does not hold for the metastable point, i.e. any small deviation from the metastable point is immediately amplified by the positive feedback and the circuit moves away from the metastable point towards one of the two stable points. Since electronic circuits are constantly affected by small deviations due to random noise, an SRAM cell will never stay in its metastable state very long but will quickly end up in one of the two stable states (randomly).

Basic Operation

The operation principle of an SRAM PUF is based on the transient behavior of an SRAM cell when it is powered up, i.e. when its supply voltage V_{DD} comes up. The circuit will evolve to one of its operating points, but it is not immediately clear to which one. The preferred initial operating point of an SRAM cell is determined by the difference in 'strength' of the MOSFETs in the cross-coupled inverter circuit. The transient behavior of an SRAM cell at power-up is shown in Fig. 2.5d. For efficiency and performance reasons, typical SRAM cells are designed to have perfectly

matched inverters. The actual difference in strength between the two inverters, the so-called *device mismatch*, is caused by random process variations in the silicon production process and is hence cell-specific. Each cell will have a random preferred initial operating point.

When one of the inverters is significantly stronger than the other one, the preferred initial operating point will be a stable state and the preference will be very distinct, i.e. such a cell will always power-up in the same stable state, but which state this is ('0' or '1') is randomly determined for every cell. When the mismatch in a cell is small, the effect of random circuit noise comes into play. Cells with a small mismatch still have a preferred initial stable state which is determined by the sign of the mismatch, but due to voltage noise there is a non-negligible probability that they power-up in their non-preferred state. Finally, cells which, by chance, have a negligible mismatch between their inverters, will power up in, or very close to, the metastable operating point. Their final stable state will be largely random for every power-up.

The magnitude of the impact of process variations on random device mismatch in SRAM cells causes most cells to have a strongly preferred but cell-specific initial state, and only few cells have a weak preference or no preference at all. This means that the power-up state of a typical SRAM cell shows strong PUF behavior. Large arrays of SRAM cells are able to provide thousands to millions of response bits for this SRAM PUF. The address of a specific cell in such an array can be considered the challenge of the SRAM PUF.

Results

SRAM PUFs were proposed by Guajardo et al. [45] and a very similar concept was simultaneously presented by Holcomb et al. [52]. Guajardo et al. [45] collect the power-up state of 8190 bytes of SRAM from different memory blocks on different FPGAs. The results show an average inter-distance between two different blocks of 49.97 %, and the average intra-distance within multiple measurements of a single block is 3.57 % at nominal conditions and at most 12 % for large temperature deviations. Holcomb et al. [52, 53] study the SRAM power-up behavior on two different platforms: a commercial off-the-shelf SRAM chip and embedded SRAM in a microcontroller chip. For 5120 blocks of 64 SRAM cells measured on eight commercial SRAM chips at nominal conditions, they obtained an average inter-distance of 43.16 % and an average intra-distance of 3.8 %. For 15 blocks of 64 SRAM cells from the embedded memory in three microcontroller chips, they obtained $\mu_{\mathcal{P}}^{\text{inter}} = 49.34$ % and $\mu_{\mathcal{P}}^{\text{intra}} = 6.5$ %.

SRAM PUFs were tested more extensively by Selimis et al. [126] on 68 SRAM devices implemented in 90 nm CMOS technology, including advanced reliability tests considering the ramp time of the supply voltage (t_{ramp}) and accelerated ageing of the circuit for an equivalent ageing time (t_{age}) of multiple years. Their test results show an average inter-distance $\mu_{\mathcal{P}}^{\text{inter}} \approx 50$ % and intra-distances of $\mu_{\mathcal{P}}^{\text{intra}} < 4$ % (nominal), $\mu_{\mathcal{P}}^{\text{intra}} < 19$ % ($T_{env} = -40$ °C), $\mu_{\mathcal{P}}^{\text{intra}} \approx 6$ % ($V_{dd} \pm 10$ %), $\mu_{\mathcal{P}}^{\text{intra}} < 10$ %

($t_{ramp} = 1$ ms) and $\mu_{\mathcal{P}}^{\text{intra}} < 14\,\%$ ($t_{age} = 4.7$ years) respectively. Schrijen and van der Leest [124] also perform a large investigation into the reliability and uniqueness of SRAM PUFs across implementations in five different CMOS technologies (from 180 nm to 65 nm) coming from different SRAM vendors. For results we refer to their paper. Both these extensive studies prove the generally strong PUF behavior of SRAM power-up values over widely varying conditions and technologies.

2.4.5 Latch, Flip-Flop, Butterfly, Buskeeper PUFs

Besides SRAM cells, there are a number of alternative, more advanced digital storage elements which are based on the bistability principle. They all have the potential to display PUF behavior based on random mismatch between nominally matched cross-coupled devices.

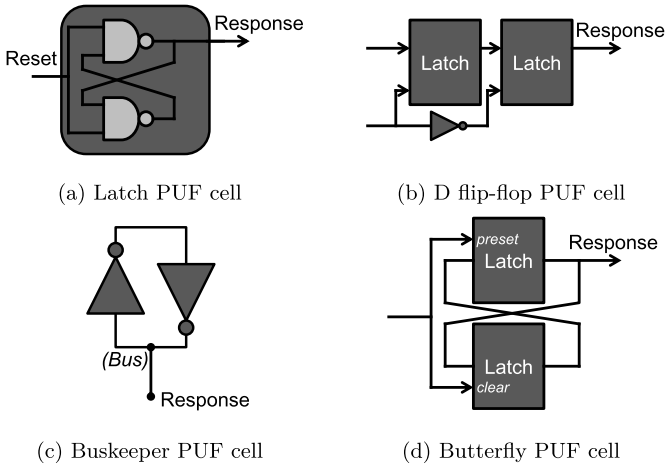
Latch PUFs

Su et al. [135] present an IC identification technique based on the settling state of two cross-coupled NOR-gates which constitute a simple SR latch. By asserting a reset signal, this latch is forced into an unstable state and when released it will converge to a stable state depending on the internal mismatch between the NOR gates. Experiments on 128 NOR-latches implemented on 19 ASICs manufactured in 130 nm CMOS technology yield $\mu_{\mathcal{P}}^{\text{inter}} = 50.55\,\%$ and $\mu_{\mathcal{P}}^{\text{intra}} = 3.04\,\%$. An equivalent latch PUF cell structure based on cross-coupled NAND gates is possible, as shown in Fig. 2.6a.

A practical advantage of this latch PUF construction over SRAM PUFs is that the PUF behavior does not rely on a power-up condition, but can be (re)invoked at any time when the device is powered. This implies that secrets generated by the PUF need not be stored permanently over the active time of the device but can be recreated at any time. Moreover, it allows us to measure multiple evaluations of each response, which makes it possible to improve reliability through post-processing techniques such as majority voting. An interesting approach is taken by Yamamoto et al. [156], who use multiple evaluations of a latch PUF on FPGA to detect unreliable cells. Instead of discarding these cells, they consider them as a third possible power-up state, besides stable at '0' and stable at '1', which effectively increases the response length of the PUF by a factor of at most $\log_2(3)$.

Flip-Flop PUFs

In [84], we propose a PUF based on the power-up behavior of clocked D flip-flops on an FPGA platform. Most D flip-flop implementations are composed of a number of latch structures which are used to store a binary state, as shown in Fig. 2.6b. These internal latch structures cause the PUF behavior of a D flip-flop as for the basic latch



(a) Latch PUF cell (b) D flip-flop PUF cell

(c) Buskeeper PUF cell (d) Butterfly PUF cell

Fig. 2.6 Different PUFs based on bistable memory elements

PUF. We measure the power-up values of 4096 D flip-flops from three FPGAs and apply simple majority voting post-processing techniques, generating one response bit from 5 measurements on 9 flip-flops, to improve the uniqueness of the responses. This yields $\mu_{\mathcal{P}}^{\text{inter}} \approx 50$ % and $\mu_{\mathcal{P}}^{\text{intra}} < 5$ %.

van der Leest et al. [150] perform a more elaborate experimental study, measuring the power-up values of 1024 flip-flops on each of 40 ASIC devices and under varying conditions. Their experiments yield an average inter-distance of 36 % on the raw flip-flop values, and an average intra-distance strictly smaller than 13 % even under large temperature variations. They also propose a number of basic post-processing techniques to increase the uniqueness of the responses, including von Neumann extraction [153] and XOR-ing response bits.

Butterfly PUFs

For FPGA platforms, SRAM PUFs are often impossible because the SRAM arrays on most commercial FPGAs (when present) are forcibly cleared immediately after power-up. This results in the loss of any PUF behavior. In an attempt to provide an SRAM PUF-like construction which works on standard FPGAs, we propose the butterfly PUF [72]. The behavior of an SRAM cell is mimicked in the FPGA reconfigurable logic by cross-coupling two transparent data latches, forming a bistable circuit depicted in Fig. 2.6d. Using the preset/clear functionality of the latches, this circuit can be forced into an unstable state and will again converge when released. Measurement results on 64 butterfly PUF cells on 36 FPGAs yield $\mu_{\mathcal{P}}^{\text{inter}} \approx 50$ % and $\mu_{\mathcal{P}}^{\text{intra}} < 5$ % under high temperature conditions. It must be noted that, due to the discrete routing options on FPGAs, it is not trivial to implement the cell in such a way that the *mismatch by design* is small. This is a necessary condition if one wants the random mismatch caused by manufacturing variability to have any effect.

Buskeeper PUFs

Simons et al. [132] propose yet another variant of a bistable memory element PUF based on buskeeper cells. A buskeeper cell is a component that is connected to a bus line on an embedded system. It is used to maintain the last value which was driven on the line and prevents the bus line from floating. Internally, a buskeeper cell is a simple latch with a weak drive-strength. A cross-coupled inverter structure at their core, as shown in Fig. 2.6c, is again the cause of the PUF behavior of the power-up state of these cells. An advantage of this PUF over other bistable memory cell PUFs is that a basic buskeeper cell is very small, e.g. in comparison to typical latches and D flip-flops.

2.4.6 Bistable Ring PUF

The bistable ring PUF as proposed by Chen et al. [22] presents a combination of elements from both ring oscillator PUFs and SRAM PUFs. Its structure is very similar to that of a ring oscillator PUF, consisting of a challengeable loop of inverting elements. However, in contrast to the ring oscillator PUF, the number of inverting elements is even, which implies that the loop does not oscillate but exhibits bistability, like the SRAM PUF. Using reset logic, the bistable ring can be destabilized and after a settling time it stabilizes to one of the two stable states. The preferred stable state is again a result of process variations and hence instance-specific and usable as a PUF response. A single loop structure can be configured in many different ways, each exhibiting its own preferred stable state, and the exact configuration is controlled by the PUF's challenge. Chen et al. [22] observe that the settling time to reach a stable state is very challenge-dependent. Evaluating a response too quickly, before the ring is completely settled, results in severely reduced uniqueness. An implementation of a 64-element bistable ring PUF was configured on eight FPGAs and evaluated 12 times for each of 50000 different challenges. With a settling time of 47 µs, this results in an average observed intra-distance of 2.19 % at nominal conditions and 5.81 % at $T_{env} = 85$ °C, and an average inter-distance of 41.91 %. Due to the structural similarities between the bistable ring and a challengeable delay chain, as for the arbiter PUF, modeling attacks could be possible. Currently, no modeling results are known, but further analysis is required before strong claims regarding the unpredictability of the bistable ring PUF can be made.

2.4.7 Mixed-Signal PUF Constructions

Next, we describe a number of integrated electronic PUF proposals whose PUF behavior is inherently of an analog nature. This means that they also require embedded analog measurement techniques, and an analog-to-digital conversion if their

responses are required in digital form, making them mixed-signal electronic constructions. While they are technically labelled as intrinsic PUFs, since they can be integrated on a silicon die and take their randomness from process variations, it is not straightforward to deploy these constructions in a standard digital silicon circuit, due to their mixed-signal nature and possibly due to non-standard manufacturing processes.

ICID: A Threshold Voltage PUF

To the best of our knowledge, Lofstrom et al. [82] were the first to propose an embedded technique called ICID which measures implicit process variations on integrated circuits and uses them as a unique identifier of the embedding silicon die. Their construction consists of a number of equally designed transistors laid out in an addressable array. The addressed transistor drives a resistive load and because of the effect of random process variations on the threshold voltages of these transistors, the current through this load will be partially random. The voltage over the load is measured and converted to a bit string with an auto-zeroing comparator. The technique was experimentally verified on 55 ASICs produced in 350 nm CMOS technology. An average intra-distance under extreme environmental variations of $\mu_{\mathcal{P}}^{\mathsf{intra}} = 1.3\ \%$ is observed, while $\mu_{\mathcal{P}}^{\mathsf{inter}}$ is very close to 50 %.

Inverter Gain PUF

Puntin et al. [106] present a PUF construction based on the variable gain of equally designed inverters, caused by process variations. The difference in gain between a pair of inverters is determined in an analog electronic measurement and converted into a single bit response. The proposed construction is implemented in 90 nm CMOS technology and an experimentally observed average intra-distance $<0.07\ \%$ at nominal conditions and $<0.4\ \%$ at ($T_{env} = 125\ ^{\circ}\mathrm{C}$, $V_{dd} + 10\ \%$) is reported; however, this already includes a masking of the 20 % least stable response bits. The inter-distance is not presented, but it is mentioned that the correlation between response strings from different PUF instances is less than 1 %.

SHIC PUFs

Rührmair et al. [117, 121] propose a circuit consisting of an addressable array of diodes which is implemented as a crossbar memory in the aluminium-induced layer exchange (ALILE) technology. They observe that the read-out time of each memory cell, consisting of a single diode, is influenced by process variations and hence usable as a PUF response. Since each crosspoint in a dense grid of metal lines creates a diode, a very high physical density of response bits of up to 10^{10} bit/cm^2 can be obtained; hence the name Super-High Information Content or SHIC PUF. The per-

formance of the construction is very low, with read-out speeds limited to merely 100 bit/s. The authors claim that the combination of high density and low performance provides protection against full read-out attacks, but it is unclear whether the particularly low performance makes any practical application possible. The physical mechanisms to produce such a SHIC PUF were tested, but no experimental PUF data is provided.

SRAM Failure PUF

An SRAM PUF variant is proposed by Fujiwara et al. [38]. The *static noise margin* (SNM) of an SRAM cell is a key figure of merit describing the resistance of a cell to voltage noise. Voltage noise exceeding the SNM alters a cell's state and hence deletes the information it is storing, resulting in a bit failure. The exact value of a cell's SNM is also influenced by silicon process variations; hence each cell in an SRAM array will produce a bit failure at a slightly different noise level. Fujiwara et al. [38] apply this observation to build a PUF. By gradually increasing the word line voltage of an SRAM array, the SNM of a cell is reduced until it produces a bit failure. They do this for a large array of cells and identify the first n failing cells. The addresses of these cells are random and are used as the PUF's responses. They present experimental results from 53 test ASICs, each producing 128 response bits, and report an average inter-distance of 49.92 %. The average observed intra-distance is as low as 0.1 %, but this is already after an elaborate majority voting post-processing. A very similar concept is proposed by Krishna et al. [70] as the MECCA PUF. They shorten the word line duty cycle of the SRAM array to induce failures.

2.4.8 Overview of Experimental Results

Table 2.1 summarizes all experimental results on intrinsic PUF constructions that are discussed in this section. We compare the average response intra- and inter-distances of the proposals since they are the most important quality parameters of a PUF and are therefore nearly always mentioned for a PUF experiment. When possible, we also mention the details of the experiment which produced these statistics, i.e. the condition under which the experiment was performed (α), and the size of the experiment in terms of number of different PUF instances (N_{puf}), number of challenges measured per instance (N_{chal}), and number of evaluations of each challenge (N_{meas}).

Although the intra- and inter-distance means give a good first notion of the quality of a PUF, there are a number of other measures which need to be considered when objectively comparing different PUF proposals. For one, the PUF proposals listed in Table 2.1 represent a wide variety of different implementations. Besides the basic PUF quality, the implementation efficiency and performance are equally important

Table 2.1 Overview of experimental results of intrinsic PUF constructions in the literature

Intrinsic PUF proposal $\mathcal{P}$	Experiment details				Experiment results	
	Condition α	N_{puf}	N_{chal}	N_{meas}	$\mu_{\mathcal{P}}^{\text{inter}}$	$\mu_{\mathcal{P};\alpha}^{\text{intra}}$
Simple Arbiter PUF [43]	nominal	23 (FPGA)	100000	200	0.10 %	1.05 %
Simple Arbiter PUF [43]	$T_{env} = 60$ °C	23 (FPGA)	100000	23	0.30 %	1.05 %
Simple Arbiter PUF [75]	nominal	37 (ASIC)	10000	?	0.70 %	23.00 %
Simple Arbiter PUF [75]	$T_{env} = 67$ °C	37 (ASIC)	10000	?	4.82 %	23.00 %
Simple Arbiter PUF [75]	$V_{dd} - 2$ %	37 (ASIC)	10000	?	3.74 %	23.00 %
FF Arbiter PUF [79]	nominal	20 (ASIC)	100000	?	4.50 %	38.00 %
FF Arbiter PUF [79]	$T_{env} = 67$ °C	20 (ASIC)	100000	?	9.84 %	38.00 %
Basic ROPUF [42]	$T_{env} = 50$ °C	4 (FPGA)	?	?	0.01 %	1.00 %
ROPUF [136]	$T_{env} = 120$ °C & $V_{dd} + 10$ %	15 (FPGA)	128	?	0.48 %	46.15 %
ROPUF [91]	nominal	125 (FPGA)	511	100	0.86 %	47.13 %
ROPUF [91]	$T_{env} = 65$ °C	5 (FPGA)	511	100	4.00 %	47.13 %
ROPUF [91]	$V_{dd} - 20$ %	5 (FPGA)	511	100	15.00 %	47.13 %
Improved ROPUF [90]	nominal	5 (FPGA)	255	?	0.00 %	44.10 %
Improved ROPUF [90]	$T_{env} = 65$ °C	5 (FPGA)	255	?	0.00 %	44.10 %
Improved ROPUF [90]	$V_{dd} - 20$ %	5 (FPGA)	255	?	2.00 %	44.10 %
Enhanced ROPUF [92]	$T_{env} = 70$ °C	125 (FPGA)	65519	?	49.99 %	10 %
Glitch PUF [1]	$T_{env} = 70$ °C	36 (FPGA)	128	?	3.60 %	48.30 %
Glitch PUF [129]	nominal	16 (FPGA)	2048	?	1.30 %	35.00 %
Glitch PUF [129]	$T_{env} = 85$ °C & $V_{dd} + 5$ %	16 (FPGA)	2048	?	15.00 %	35.00 %

Table 2.1 (*Continued*)

Intrinsic PUF proposal $\mathcal{P}$	Experiment details				Experiment results	
	Condition α	N_{puf}	N_{chal}	N_{meas}	$\mu_{\mathcal{P}}^{\text{inter}}$	$\mu_{\mathcal{P};\alpha}^{\text{intra}}$
SRAM PUF [45]	nominal	? (FPGA)	8190	92	3.57 %	49.97 %
SRAM PUF [45]	$T_{env} = 80$ °C	? (FPGA)	8190	?	12.00 %	49.97 %
SRAM PUF [53]	nominal	5120 (COTS)	64	?	3.80 %	43.16 %
SRAM PUF [53]	nominal	15 (μC)	64	?	6.50 %	49.34 %
SRAM PUF [126]	nominal	68 (ASIC)	2048	?	<4 %	≈50.00 %
SRAM PUF [126]	$T_{env} = -40$ °C	68 (ASIC)	2048	?	<19 %	≈50.00 %
SRAM PUF [126]	$V_{dd} \pm 10$ %	68 (ASIC)	2048	?	6.00 %	≈50.00 %
SRAM PUF [126]	$t_{ramp} = 1$ ms	68 (ASIC)	2048	?	<10 %	≈50.00 %
SRAM PUF [126]	$t_{age} = 14$ years	68 (ASIC)	2048	?	<14 %	≈50.00 %
Latch PUF [135]	nominal	19 (ASIC)	128	?	3.04 %	50.55 %
Flip-Flop PUF [84]	nominal	3 (FPGA)	4096	?	<5 %	≈50.00 %
Flip-Flop PUF [150]	$T_{env} = 80$ °C	40 (ASIC)	1024	?	<13 %	36.00 %
Butterfly PUF [72]	$T_{env} = 80$ °C	36 (FPGA)	64	?	<5 %	≈50.00 %
Bistable Ring PUF [22]	nominal	8 (FPGA)	50000	12	2.19 %	41.91 %
Bistable Ring PUF [22]	$T_{env} = 85$ °C	8 (FPGA)	50000	12	5.81 %	41.91 %
ICID [82]	nominal	55 (ASIC)	132	?	1.30 %	≈50.00 %
SRAM Failure PUF [38]	nominal	53 (ASIC)	128	?	0.01 %	49.92 %

for most applications. However, for many PUF constructions, detailed implementation parameters are not provided, or implementation efficiency is even sacrificed to improve the PUF quality. This gives a distorted picture when comparing different PUFs solely based on Table 2.1. Moreover, the listed PUF constructions are also implemented on a variety of platforms and technologies. We differentiate the used platforms between field-programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), commercial off-the-shelf products (COTS) and microcontrollers (μCs). However, even within a single platform, different technologies can be used, e.g. the different scaling CMOS technology nodes. Finally, as is clear from the experiment details, the size of the experiments and hence the statistical significance of the obtained results also varies greatly. To overcome these issues and provide an objective comparison between intrinsic PUFs, we present an extensive experimental analysis of a number of intrinsic PUF constructions in Chap. 4.

2.5 PUF Extensions

2.5.1 POKs: Physically Obfuscated Keys

The concept of a physically obfuscated key or POK was introduced by Gassend [40], and generalized to physically obfuscated algorithms by Bringer et al. [14]. The only condition for a POK is that a key is permanently stored in a 'physical' way instead of a digital way, which makes it hard for an adversary to learn the key by a probing attack. Additionally, an invasive attack on the device storing the key should destroy the key and make further use impossible, hence providing tamper evidence. It is clear that POKs and PUFs are very similar concepts, and it has already been pointed out by Gassend [40] that POKs can be built from (tamper-evident) PUFs and vice versa.

2.5.2 CPUFs: Controlled PUFs

A controlled PUF or CPUF, as introduced by Gassend et al. [41], is in fact a mode of operation for a PUF in combination with other (cryptographic) primitives. A PUF is said to be controlled if it can only be accessed via an algorithm which is physically bound to the PUF in an inseparable way. Attempting to break the link between the PUF and the access algorithm should preferably lead to the destruction of the PUF. There are a number of advantages by turning a PUF into a CPUF:

- A (cryptographic) hash function to generate the challenges of the PUF can prevent chosen-challenge attacks, e.g. to make model-building attacks more difficult. However, for arbiter PUFs it has been shown that model-building attacks work equally well for randomly picked challenges.

- An error-correction algorithm acting on the PUF measurements makes the final responses much more reliable, reducing the probability of a bit error in the response to virtually zero.
- A (cryptographic) hash function applied on the error-corrected outputs effectively breaks the link between the responses and the physical details of the PUF measurement. This makes model-building attacks much more difficult. When hashing a PUF's response, error-correction is indispensable since any minor deviation on the response gives an entirely unrelated hash result.
- The hash function generating the PUF challenges can take additional inputs, e.g. allowing to give a PUF multiple personalities. This might be desirable when the PUF is used in privacy-sensitive applications.

2.5.3 RPUFs: Reconfigurable PUFs

Reconfigurable PUFs or RPUFs were introduced by Kursawe et al. [73]. The idea behind an RPUF is to extend the regular challenge-response behavior of a PUF with an additional operation called *reconfiguration*. This reconfiguration has the effect that the partial or complete challenge-response behavior of the PUF is randomly and preferably irreversibly changed, leading to a new PUF. Kursawe et al. [73] propose two possible implementations of RPUFs where the reconfiguration mechanism is an actual physical reconfiguration of the randomness in the PUF. One is an extension of optical PUFs, where a strong laser beam briefly melts the optical medium, causing a random rearrangement of the optical scatterers, which leads to a completely new optical challenge-response behavior. The second proposal is based on a type of non-volatile storage called phase change memory. Writing to such a memory amounts to physically altering the phase of a small cell from crystalline to amorphous or somewhere in between, and it is read out by measuring the resistance of the cell. Since the resistance measurements are more accurate than the writing precision, the exact measured resistances can be used as responses, and rewriting the cells will change them in a random way. Both proposals are rather exotic at this moment and remain largely untested. A third proposed option is actually a logical extension of a regular PUF. By fixing a part of a PUF's challenge with a fuse register, the PUF can be reconfigured by blowing a fuse, which optimally leads to a completely changed challenge-response behavior for the remaining external challenge bits.

Katzenbeisser et al. [64] generalize this last option in a logically reconfigurable PUF (LRPUF). An LRPUF consists of a regular PUF, a portion of non-volatile memory that stores a state and state-dependent input- and output-transformations respectively on the PUF's challenge and response. By updating the state of an LRPUF using an appropriate state-update mechanism, the input- and output-transformations change, resulting in a completely different challenge-response behavior for the whole construction. This state update is hence a logical form of reconfiguring the PUF. Katzenbeisser et al. [64] propose a number of practical input and output transformations and state update mechanisms which achieve interesting security properties. LRPUFs have some drawbacks with respect to actual *physically reconfigurable*

PUFs as proposed by Kursawe et al. [73]: (i) it is debatable whether the reconfiguration of an LRPUF is truly irreversible, since if an old internal state is restored, the same challenge-response behavior will reappear, and (ii) the security of the logical reconfiguration relies on the integrity of the state memory, which needs to be guaranteed by independent physical protection measures. However, their relatively easy construction in comparison to the rather exotic physically reconfigurable PUF proposals makes them immediately deployable.

A more specific form of a reconfigurable PUF is introduced by Rührmair et al. [120] as an *erasable PUF*. An erasable PUF is best described as a reconfigurable PUF with a reconfiguration granularity of a single challenge-response pair, i.e. the behavior of a single challenge-response pair of an erasable PUF instance can be (irreversibly) altered while keeping the behavior of all other pairs fixed. As demonstrated by Rührmair et al. [120], this construction enables some interesting security features for protocols based on erasable PUFs.

2.5.4 PPUFs: Public PUFs and SIMPL Systems

A number of attempts to use PUFs as part of a public-key-like algorithm have been made. SIMPL systems were proposed by Rührmair [112] and are an acronym for *Simulation Possible but Laborious*. Two potential implementations of such a system are discussed by Rührmair et al. [114]. A very similar concept was proposed by Beckmann and Potkonjak [6] as Public PUFs or PPUFs. Both SIMPL systems and PPUFs rely on physical challenge-response systems (PPUFs) which can be modeled explicitly, but for which evaluating the model is laborious and takes a detectably longer amount of time than the evaluation of the PPUF itself.

2.6 Conclusion

2.6.1 Overview of PUF Constructions

In this chapter, we have extensively studied the physically unclonable function concept through an elaborate overview and discussion of known constructions. This overview provides a deep exploration of the great diversity of different PUF proposals, each with its own implementation details, practical considerations, security characteristics and performance results. The listing and comparison of these features, which by itself already serves as a convenient reference work, will be of great value when trying to determine a set of common defining properties of PUFs in Chap. 3. We also propose an interesting classification of PUFs into intrinsic and non-intrinsic based on certain practical qualities of their construction. It is argued why intrinsic PUFs are considered advantageous with regard to security and cost-efficiency. In the presented overview, as is the case in PUF-related literature, the focus is therefore on intrinsic PUFs.

2.6.2 Insight into PUF Constructions

The presented overview also provides a lateral insight into PUF implementation techniques and their effect on the quality of the PUF behavior. When we focus on the intrinsic PUF proposals, a number of interesting observations can be made:

- Most intrinsic PUF constructions deploy a *differential measurement* technique, e.g. the arbiter PUF and the ring oscillator PUF, and all of the PUFs based on bistability. This turns out to be beneficial, both for the uniqueness and the reliability of their responses. Considering uniqueness, a (small) differential structure is often much more strongly affected by the locally random influence of process variations, and less prone to exhibit bias due to deterministic variations which mostly manifest at larger scales. For reliability, it is wellknown that a differential measurement exhibits a much lower sensitivity to measurement noise and is able to even completely compensate the effect of some external conditions on a measured value.
- For a given PUF construction, the quality of its PUF behavior can be enhanced in one of two ways to amplify uniqueness and reduce noise: (i) by physically tweaking the implementation details, and (ii) by algorithmically post-processing the measured values into responses.
- For physical enhancements of a given PUF construction, it is a rule of thumb that the quality of a construction's PUF behavior is directly influenced by the degree of low-level physical control one has over the implementation technology. The reasoning behind this is that the source of the PUF behavior is always of a purely physical nature, and the closer to bare physics one can design a construction, the more possibilities one has of accurately capturing this behavior. This is clear from the results on the mixed-signal-based intrinsic PUF constructions in Sect. 2.4.7: by evaluating the PUF behavior at the analog level, one is able to build more reliable and more unique PUFs. However, on the down side, the development effort also rises exponentially when designing an implementation at increasingly lower levels of physical abstraction. Additionally, many standard manufacturing flows do not provide low levels of physical manufacturing detail.
- Simple algorithmic improvements such as masking of unreliable responses and majority voting over many responses to improve reliability and/or uniqueness are in some proposals considered an inherent part of the PUF construction. As will become clear in the following chapters, algorithmic post-processing of PUF responses, to make them fit for being used in an application, can take very elaborate forms and is typically not considered as part of the PUF construction, but as a separate primitive. Algorithmic improvements nearly always imply a trade-off between the efficiency and/or performance of the PUF construction and the quality of its output, e.g. to obtain a number of PUF-derived bits with a very high reliability level, the required number of actual PUF response bits needed can be tenfold or even much more.

http://www.springer.com/978-3-642-41394-0

Physically Unclonable Functions
Constructions, Properties and Applications
Maes, R.
2013, XVII, 193 p. 28 illus., Hardcover
ISBN: 978-3-642-41394-0