Chapter 5
PUF-Based Entity Identification and Authentication

5.1 Introduction

5.1.1 Motivation

Due to its combination of uniqueness and reproducibility, a PUF embedded by an entity serves as an identifying feature of that entity, as already intuitively expressed by Definition 9. Moreover, the physical unclonability exhibited by an embedded PUF construction provides even strong security guarantees regarding this expressed identity, which could be used for authentication purposes. However, in order to be of any practical value, the security and robustness of a PUF-based identification or authentication needs to be quantified based on its experimentally verified behavioral characteristics.

Entity Authentication

In information security, the term 'authentication' has a very broad meaning, which often leads to confusion when not described in more detail. First of all, authentication can relate to entities or to data. In the former case one speaks of *entity authentication*, while the latter is called *message authentication*. Since a PUF provides a measure of an entity-specific physical feature, we particularly consider entity authentication in this chapter. Whenever we talk about PUF-based authentication, entity authentication is implied.

Entity authentication by itself is still a catchall for a collection of techniques used to check and be assured of the identity of an entity. The Handbook of Applied Cryptography [96] defines entity authentication as:

Definition 25 An entity authentication technique assures one party, through acquisition of corroborative evidence, of both: (i) the *identity* of a second party involved, and (ii) that the second party was *active* at the time the evidence was created or acquired.

Besides giving convincing proof of its identity, an entity authentication technique also needs to guarantee that the authenticating entity is actively present in the authentication.

Identification

The Handbook of Applied Cryptography [96] treats 'identification' and 'entity authentication' as synonyms. However, in this book, as in many other treatises of the subject, we consider identification to be a related but significantly weaker concept than authentication (see also [96, Remark 10.2]). *Identification* is the mere claiming or stating of its identity, without necessarily presenting any convincing proof thereof. While not strictly a security technique since it doesn't fulfill any meaningful security objective, identification still has very useful qualities:

- Identification is in many cases a necessary precondition for entity authentication and hence an inherent part of most entity authentication techniques. An entity that cannot be identified, cannot be individually authenticated.[1]
- For applications without strict security objectives, identification can be a sufficient condition, e.g. for applications which involve the tracking of products in a closed system.
- In certain situations, identification is sufficient to achieve entity authentication, since the authentication conditions are implicitly met.

For this reason, we will first discuss PUF-based identification in this chapter, before we treat PUF-based authentication.

5.1.2 Chapter Goals

The primary goal of this chapter is to propose practical methods for achieving entity identification and authentication based on the uniqueness and unpredictability of a PUF's challenge-response behavior, and introduce a methodology for quantifying the resulting identification and authentication performance in terms of security and robustness. More specifically, we aim to:

- Study how to use an entity's inherent PUF responses as an identifying feature in an identification system, and how this relates to classic identification based on assigned identities.
- Derive performance metrics for such a PUF-based identification system and apply these on the experimentally derived intrinsic PUF characteristics from Chap. 4, to provide an objective comparison of the identification performance and efficiency of the studied PUFs.
- Develop an entity authentication protocol which: (i) uses a PUF's unique and unpredictable responses directly as an authentication secret, (ii) can be deployed

[1] In an anonymous credential scheme, an entity can prove its group membership without revealing its individual identity.

based on existing intrinsic PUFs, and (iii) is sufficiently lightweight to be implemented on resource-constrained devices.
- Derive the performance metrics of the developed authentication scheme and equivalently apply them to the experimental intrinsic PUF characteristics from Chap. 4 to make an objective comparison of their authentication performance.

5.1.3 Chapter Overview

How to safely and reliably identify an entity based on its inherent PUF responses is discussed and analyzed in Sect. 5.2, and a performance overview of the different intrinsic PUFs studied in Chap. 4 is given. In Sect. 5.3, we first describe the operation of an earlier proposed basic PUF-based challenge-response authentication scheme and point out its perceived shortcomings. Based on this analysis, we propose a new and more practical PUF-based mutual authentication scheme and study its authentication performance. Finally, we conclude this chapter in Sect. 5.4.

5.2 PUF-Based Identification

5.2.1 Background: Assigned Versus Inherent Identities

When we compared PUFs to human fingerprints in Sect. 2.1.1, we introduced the concept of an *inherent* identifying feature, i.e. an entity-specific characteristic that arises in the creation process of the entity. As opposed to inherent identities, an entity can also have assigned identities. In the analogy with human beings, this is the distinction we make between fingerprints, which are inherent, and, e.g. a person's name, which is 'assigned' after birth. The inherency of its instance-specific behavior was indicated as one of the key conditions for a construction to be called a PUF.

An inanimate object can also have an assigned identity, e.g. a unique serial number or barcode which is printed on its surface, and in most applications that require entity identification, assigned identities are currently standard practice. In particular for digital silicon chips, unique bit strings which are programmed in a non-volatile memory embedded on the chip were until recently the only way of identifying a specific chip in a digital interaction. With the introduction of silicon PUF technology, it is now possible to also use inherent unique features of a silicon chip for instance identification.

Identity Provisioning Versus Enrollment

Identification techniques based on assigned as well as on inherent identities typically work in two phases. The first phase is different for both types:

- For *assigned identities*, the first phase of any identification technique consists of providing every entity that needs to be identified with a permanent unique identity. We call this the *provisioning phase*.

- For *inherent identities*, the first phase of any identification technique consists of collecting the inherent identities of every entity that needs to be identified. We call this the *enrollment phase*.

The second phase is very similar for both types and consists of an entity presenting its identity, either assigned or inherent, when requested. This is called the *identification phase*.

Practical Advantages of Inherent Identities

The differences between provisioning, for assigned identities, and enrollment, for inherent identities, highlight interesting practical advantages of the latter:

- A unique assigned identity needs to be generated before it is assigned to an entity. To ensure that all generated identities are unique (with high probability), the provisioning party either needs to keep state, e.g. using a monotonic counter, or requires a randomness source, e.g. a true random-number generator. For inherent identities this is not required, since their uniqueness results from the creation process of the entities.
- When assigning an identity to an entity, the provisioning party needs to make permanent (or at least non-volatile) physical changes to each entity. This needs to be supported by the entity's construction. In particular for silicon chips, the inclusion of a non-volatile digital memory can induce a non-negligible additional cost. Evidently, inherent identities do not require additional physical storage capabilities.
- Enrollment (reading an identity) is generally less intrusive than provisioning (writing an identity); hence it can be done faster and with a higher reliability. This is of particular interest for entities created in high-volume manufacturing flows (like silicon chip products), where unit cost is directly affected by the yield and processing time of each manufacturing step.

On the downside, there is no direct control over the actual values taken by inherent identifiers. This is an issue if one wants to assign a meaning to an identifier value, e.g. a serial number which is based on an entity's creation date. For assigned identities, one has absolute control over the identifier values. Another peculiarity of most inherent identities is their so-called *fuzzy nature*, which we discuss next.

5.2.2 *Fuzzy Identification*

Fuzzy Nature of Inherent Identifiers

A particular trait of most types of inherent identifying features which needs to be dealt with is that they show *fuzzy* random behavior.[2] We say that a random vari-

[2]When we use the term 'fuzzy' in this book, we relate to the notion of fuzziness as introduced by Juels and Wattenberg [61] to describe fuzzy commitment, which was later extended to fuzzy

able, like a PUF response or a biometric feature, shows fuzzy behavior if: (i) it is not entirely uniformly distributed, and (ii) it is not perfectly reproducible when measured multiple times. For PUF responses, both fuzzy characteristics are caused by the physical nature of their generation. Random physical processes that introduce entity-specific features during manufacturing are typically not uniformly distributed. Also, as already discussed in detail in Sect. 3.2.2, the response evaluation mechanisms of a PUF construction are subject to physical noise and environmental conditions which cause a non-perfect reproducibility of a PUF response value.

Assigned identities on the other hand are typically not fuzzy. The provisioning party can make sure that the assigned identities are generated from a uniform distribution. Also, once provisioned, an entity can typically reproduce its assigned identity with near-perfect reproducibility.

Fuzzy Identification with a Threshold

The fuzziness of a PUF response is most clearly depicted by its inter- and intra-distance distributions. When we consider binary response vectors and fractional Hamming distance as a distance metric, then perfectly uniformly random responses would have an expected inter-distance of exactly 50 %. It is clear from our literature overview of intrinsic PUF results in Sect. 2.4.8, as well as from our summarized experimental results on intrinsic PUFs in Sect. 4.5, that none of the existing intrinsic PUF constructions meet this condition, although some have an average inter-distance very close to 50 %. Equivalently, no intrinsic PUF exhibits perfect reproducibility with a fixed intra-distance of 0 %, and many are only reproducible up to an average intra-distance of 10 % or even more.

To assess the extent to which a PUF response can be used as an inherent identifier, we need to take its fuzziness into account. This is where the earlier discussed PUF property of *identifiability* comes into play (cf. Definition 9). We defined a PUF class to exhibit identifiability if with high probability its responses' intra-distances are smaller than their inter-distances. In this section, we make this intuitive definition very tangible, by computing the identifying power of a PUF's responses based on their intra- and inter-distance distributions.

Figure 5.1 shows an example of an estimated distribution of the inter- and intra-distances of a PUF's response. For this example, we consider a D flip-flop PUF which produces a 16-bit response. As an estimate for both distributions, we assume a binomial distribution with parameters the binomial probability estimators $\hat{p}_{\mathcal{P}}^{\text{inter}}$ and $\hat{p}_{\mathcal{P}}^{\text{intra}}$ resulting from the experimental analysis from Chap. 4 and summarized in Sect. 4.5. The process by which we computed these estimators guarantees that the assumed binomial distributions provide an accurate estimation, in particular for the

vaults by Juels and Sudan [60] and finally to fuzzy extractors by Dodis et al. [32, 33]. We are *not* referring to the homonymous but unrelated use of the word 'fuzzy' as used in fuzzy logic and fuzzy set theory. To avoid any confusion, we will also not use the ambiguous term fuzzy random variable in this context.

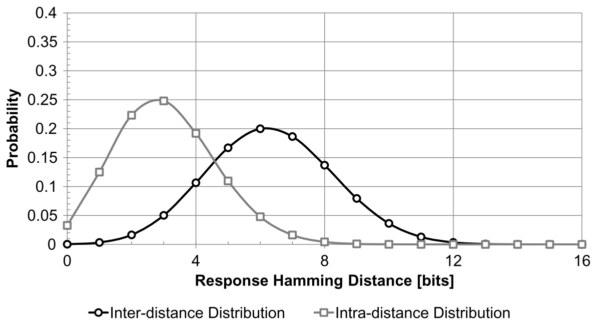


Fig. 5.1 Example: estimated inter- and intra-distance distributions for 16-bit responses from the D flip-flop PUF

right tail of the intra-distance distribution and for the left tail of the inter-distance distribution. As will become clear, this is specifically the region of interest for most applications.

From Fig. 5.1, it is clear that this PUF construction exhibits some level of identifiability, since the expected intra-distance is noticeably smaller than the expected inter-distance. However, there is also a significant overlap between the curves of the two distributions, which points out the issue of identification based on fuzzy responses. When an observed distance between responses from the enrollment and the identification phase falls in this overlapping region, it can be a result of intra-distance, in which case it is the same entity, or of inter-distance, in which case it concerns a different entity, and there is no way of distinguishing between the two cases. In a practical identification system for fuzzy identities, one needs to determine a rather pragmatic response distance threshold. Distances smaller or equal to this threshold are assumed to be intra-distances between responses from a single entity, while distances above this threshold are assumed to be inter-distances between responses from different entities. We call this threshold the *identification threshold*.

False Acceptance, False Rejection, and Equal Error Rates

During the identification phase of a PUF-based identification system, the generated response of an entity is checked against a list of enrolled responses. When an enrolled response is found whose distance from the presented response is smaller than or equal to the identification threshold, then the entity is identified as the matching entry in the list. It is clear that a fuzzy identification system based on such a pragmatic identification threshold is not 100 % reliable, especially when there is a large overlap between inter- and intra-distance distribution, as for the example in Fig. 5.1. When comparing a presented entity response to a response from the enrollment list, four possible situations can arise:

1. The presented entity is the same entity that produced the enrolled response, and manages to reproduce the enrolled response with an intra-distance smaller than the identification threshold. The presented entity is correctly identified. This is called a *true acceptance*.

2. The presented entity is the same entity that produced the enrolled response, but is not able to reproduce the enrolled response with an intra-distance smaller than the identification threshold. The presented entity is mistakenly rejected. This is called a *false rejection*.
3. The presented entity is not the same entity that produced the enrolled response, but happens (by chance) to produce a response whose inter-distance to the enrolled response is smaller than the identification threshold. The presented entity is mistakenly identified. This is called a *false acceptance*.
4. The presented entity is not the same entity that produced the enrolled response, and it produces a response whose inter-distance is larger than the identification threshold. The presented entity is correctly rejected. This is called a *true rejection*.

It is clear that false rejections and false acceptances are both undesirable for a practical identification system. The probability that a random identification attempt results in one of these cases is respectively expressed as the *false rejection rate* or FRR and as the *false acceptance rate* or FAR of the system. FAR expresses the security of an identification system, since a low FAR means that there is little risk of misidentification which could lead to security issues. FRR on the other hand expresses the robustness or usability of a system, as it expresses the risk of wrongfully rejecting legitimate entities, which would be very impractical. For a usable identification system, both FAR and FRR need to be as small as possible, but it is evident that they cannot be both minimized at the same time. As is often the case, an acceptable trade-off between security and usability needs to be made.

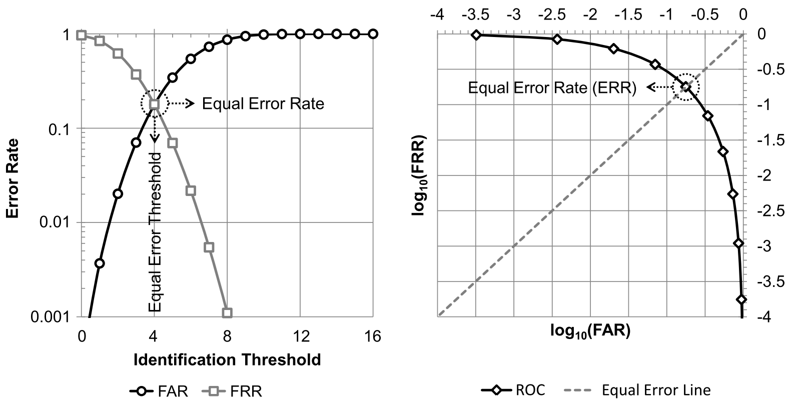
For a given identification system, FAR and FRR depend on the choice for the identification threshold value, which we denote as t_{id}. A high threshold minimizes the risk of a false rejection but increases the likelihood of false acceptances, and vice versa for a low threshold. When the distributions of the inter- and intra-distances of the considered PUF are known, the respective relations between FAR, FRR and t_{id} can be computed:

- FAR is the probability that the inter-distance is smaller than or equal to t_{id}. This is equivalent to the evaluation of the cumulative distribution function of the inter-distance at t_{id}.
- FRR is the probability that the intra-distance is larger than t_{id}. This is equivalent to the complement of the evaluation of the cumulative distribution function of the intra-distance at t_{id}.

For the example presented in Fig. 5.1, we have assumed a binomial distribution for both inter- and intra-distances; hence FAR and FRR become:

$$\mathsf{FAR}(t_{\mathsf{id}}) = F_{\mathsf{bino}}\big(t_{\mathsf{id}}; 16, \hat{p}_{\mathcal{P}}^{\mathsf{inter}}\big),$$
$$\mathsf{FRR}(t_{\mathsf{id}}) = 1 - F_{\mathsf{bino}}\big(t_{\mathsf{id}}; 16, \hat{p}_{\mathcal{P}}^{\mathsf{intra}}\big),$$

with $F_{\mathsf{bino}}(t; n, p)$ the cumulative binomial distribution function with parameters n and p evaluated in t, and $\hat{p}_{\mathcal{P}}^{\mathsf{inter}}$ and $\hat{p}_{\mathcal{P}}^{\mathsf{intra}}$ the binomial estimators for the D flip-flop



(a) FAR and FRR plots (b) ROC curve plot

Fig. 5.2 Example: identification metrics for a threshold identification system based on the PUF described by Fig. 5.1

PUF taken from Table 4.10. The resulting FAR and FRR for every identification threshold value between 0 and 16 are plotted in Fig. 5.2a. From this figure it is clear that there is a threshold, and a corresponding error rate, where the plots of FAR and FRR intersect. We call this the *equal error threshold* t_{EER} and the corresponding error rate the *equal error rate* or EER. For discrete distributions, FAR and FRR will never be exactly equal for a discrete threshold, and in that case t_{EER} and EER are defined as:

$$t_{\mathsf{EER}} \triangleq \operatorname{argmin}_t \left\{ \max \left\{ \mathsf{FAR}(t), \mathsf{FRR}(t) \right\} \right\},$$

and

$$\mathsf{EER} \triangleq \max \left\{ \mathsf{FAR}(t_{\mathsf{EER}}), \mathsf{FRR}(t_{\mathsf{EER}}) \right\}.$$

The equal error rate is also indicated in Fig. 5.2a.

When designing a PUF-based identification system, the FAR and FRR plots as shown in Fig. 5.2a can be used to find a suitable trade-off meeting the application requirements. This can, but does not need to be the EER, e.g. in some applications more care is given to security than to usability, or vice versa. A more convenient way for assessing the FRR-vs-FAR trade-off is the plot of FRR as a function of FAR as shown in Fig. 5.2b. Such a plot is also called the *receiver-operating characteristic* or ROC plot of the system. In a ROC plot, the EER is found as the intersection with the identity function which we have labelled the *equal error line* in Fig. 5.2b. ROC curves completely summarize the identification performance of an identification system and are particularly useful for comparing the performance of different systems. A more condensed performance qualifier of a particular identification system is given by its EER value.

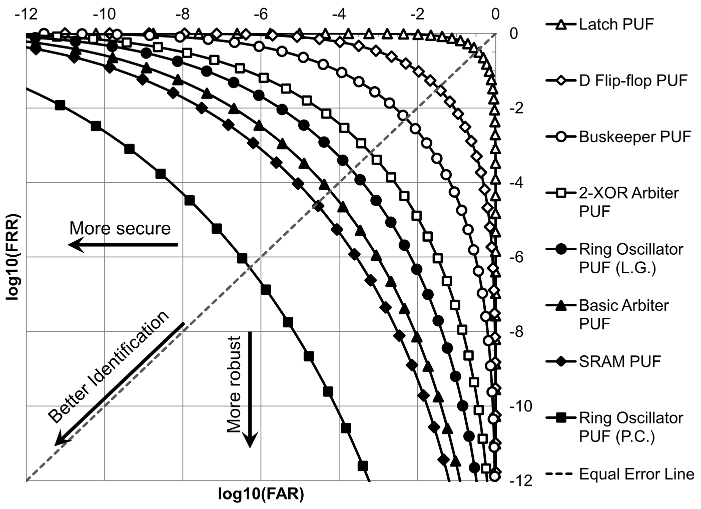


Fig. 5.3 Comparison of ROC curves for identification systems based on 64-bit responses from each of the eight experimentally verified intrinsic PUFs which are summarized by Table 4.10

5.2.3 Identification Performance for Different Intrinsic PUFs

In Sect. 5.2.2 we presented a toy example of a 16-bit D flip-flop PUF identification system and used it to introduce the different performance metrics of fuzzy identification systems. Now, we will use these introduced metrics to objectively compare the identification performance of the different intrinsic PUFs which were experimentally studied in Chap. 4. As in Sect. 5.2.2, we will assume a binomial distribution for both the inter- and intra-distances of these intrinsic PUFs and use the binomial probability estimators as summarized in Table 4.10. This assumption is justified by the fact that these estimators where derived to accurately describe the critical region of both distributions, i.e. the part where both probability mass functions overlap.

We will first compare the identification performance of all considered PUFs for a fixed length response. Afterwards, we compare the required parameters of the different PUFs in order to obtain the same identification performance. Ultimately, we combine these obtained parameters with each PUF's area estimation to generate an objective as possible comparison of identification performance versus silicon area use for each of the intrinsic PUF constructions.

Comparison of ROC Curves for 64-bit Identification with Different PUFs

We consider an identification system based on 64-bit PUF responses and apply the methods introduced in Sect. 5.2.2 to derive the ROC curves for all eight considered intrinsic PUFs, based on their parameters from Table 4.10. All eight ROC curves are plotted on the same graph in Fig. 5.3.

When reading a ROC curve, it is important to know that the more one moves to the left (up) on a curve, the more secure an identification system becomes, i.e. the less likely that a misidentification will happen. On the other hand, moving down (to the right) on a ROC curve gives more robust systems, i.e. systems which are less likely to result in an unjustified rejection. In this respect, the closer to the lower left corner a particular ROC curve is situated, the better its overall identification performance and the easier to make a meaningful security-usability trade-off.

Analyzing the ROC curves from Fig. 5.3, it is clear that an identification system based on 64-bit responses from the ring oscillator PUF with pairwise comparison greatly outperforms all the other PUFs and is the only PUF which obtains an $\mathsf{EER} \le 10^{-6}$. Looking at Table 4.10, the high identification performance of this ring oscillator PUF is caused by a combination of having nearly the highest inter-distance parameter (second to the 2-XOR arbiter PUF) and by far the best intra-distance parameter. The ROC curves of the SRAM PUF and the basic arbiter PUF follow at a considerable distance, both reaching an $\mathsf{EER} \le 10^{-4}$. Notable is the fact that the 2-XOR arbiter PUF, while having a better inter-distance parameter than the basic arbiter PUF, performs significantly worse due to its much worse intra-distance behavior. At the bottom of the ranking we find the latch PUF which performs very weakly. It does not even reach an $\mathsf{EER} \le 10\ \%$, which means that over one in ten identification attempts will either be rejected or misidentified. In fact, looking at the inter- and intra-distance parameters of the latch PUF in Table 4.10, we may conclude that the latch PUF hardly exhibits identifiability (and can hence only barely be called a PUF), since its average intra-distance is only slightly smaller than its average inter-distance.

Comparison of PUF Parameters and Areas for Practical Identification Requirements

The required identification performance of an identification system is determined by its application, but for most practical applications FAR and FRR both $\le 10^{-6}$, and hence $\mathsf{EER} \le 10^{-6}$, is minimally desired. For many applications, EER even needs to be considerably smaller, e.g. $\mathsf{EER} \le 10^{-9}$ or even $\mathsf{EER} \le 10^{-12}$ can be required for critical systems. Aiming for a lower EER also provides more freedom in selecting an optimal FAR-vs-FRR trade-off. The results from Fig. 5.3 show that with a 64-bit PUF response, only the pairwise-comparison ring oscillator PUF achieves $\mathsf{EER} \le 10^{-6}$. In order to obtain a better identification performance based on the same PUF, longer responses need to be considered as identifiers. In the following we examine which response lengths, denoted as n_{id}, are needed for every considered PUF to reach an identification performance of respectively $\mathsf{EER} \le 10^{-6}$, $\mathsf{EER} \le 10^{-9}$ and $\mathsf{EER} \le 10^{-12}$.

To ultimately obtain an objective comparison of the identification performance of the different considered PUFs, we also need to take their silicon area efficiency into account. After determining the minimal required response length n_{id} to achieve a certain identification performance, we estimate the required silicon area a PUF construction needs to occupy in order to produce a response of that length. These area estimations are based on the area breakdown of our test chip as presented in Table 4.1. We use the following approach for scaling the area of the different PUFs:

- For the memory-based PUFs, we scale the area completely bitwise, i.e. the estimated silicon area of the entire PUF block as presented in Table 4.1 is divided by the total number of bit cells of the considered PUF implemented on the test chip and multiplied by n_{id}. This is a very rough estimation as it does not take fixed overhead into account, and it assumes bit cells can be instantiated one by one. Especially for the SRAM PUF this is not very accurate since it neglects the overhead of the address decoder, sense amplifiers, readout circuitry, etc. Moreover, typical SRAM array implementations come in multiples of kilobytes, not bits. However, given the limited knowledge we have about the implemented PUF areas, this is the best estimate we can give. For the other memory-based PUFs the estimate is better since they can be instantiated on a bit by bit basis. For the D flip-flop PUF with the chained read-out implementation, this estimation is even fairly accurate.
- For both arbiter-based PUFs, the silicon area is relatively independent of the required number of response bits, since a single 64-bit arbiter PUF can technically produce 2^{64} response bits. It will become clear that for most applications, large problems arise when too many response bits of a single arbiter-based PUF are used, but for plain identification (no authentication) this is not an issue. This means that the identification performance of a single arbiter PUF is virtually unlimited. The area of a single basic arbiter PUF is estimated by dividing the estimated area of the entire arbiter PUF block as reported in Table 4.1 by the total number of instantiated arbiter PUFs on the test chip. The estimated area of the 2-XOR arbiter PUF is that of two basic arbiter PUFs.
- For both ring oscillator PUFs, responses are bit vectors which are computed from 16 simultaneously measured ring oscillator frequencies. The pairwise comparison method produces 8 bits per response, and the Lehmer-Gray method 49 bits. Multiples of these response lengths are obtained by incrementing the number of oscillators in each of the 16 batches by the same amount. The required area for these ring oscillator PUFs is estimated by dividing n_{id} by 8 and 49 respectively, and multiplying the ceiled outcome of this division with the area of a set of 16 oscillators. The area of a single oscillator is estimated by dividing the total area of the ring oscillator block from Table 4.1 by the number of oscillators implemented on the test chip. This estimation is not very accurate as it neglects the overhead of the frequency counters.

The results of these estimations are presented in Tables 5.1, 5.2 and 5.3, respectively for $\mathsf{EER} \leq 10^{-6}$, $\mathsf{EER} \leq 10^{-9}$ and $\mathsf{EER} \leq 10^{-12}$. We want to point out again that the reported silicon area results need to be considered as rough estimates at best, and even better as indications of order of magnitude.

From Tables 5.1 to 5.3 we conclude that, even though only based on rough estimations, SRAM PUFs exhibit by far the best area efficiency for a required identification performance, being an order of magnitude smaller than the next best PUF. This is due to a combination of their relatively strong PUF behavior, and in particular

Table 5.1 Comparison of PUF parameters and estimated silicon area for an identification system with $\mathsf{EER} \leq 10^{-6}$

PUF class	n_{id}	t_{id}	$\log_{10}\mathsf{FAR}$	$\log_{10}\mathsf{FRR}$	Silicon area (μm^2)
SRAM PUF	89	21	−6.00	−6.03	72.3
Latch PUF	9344	2664	−6.00	−6.01	77562.5
D Flip-Flop PUF	448	128	−6.05	−6.16	5359.4
Buskeeper PUF	223	72	−6.03	−6.05	1034.4
Arbiter PUF (basic)	101	23	−6.13	−6.21	1089.8
Arbiter PUF (2-XOR)	142	42	−6.06	−6.12	2179.7
Ring Oscillator PUF (P.C.)	62	12	−6.06	−6.20	7531.3
Ring Oscillator PUF (L.G.)	121	30	−6.12	−6.24	2824.2

Table 5.2 Comparison of PUF parameters and estimated silicon area for an identification system with $\mathsf{EER} \leq 10^{-9}$

PUF class	n_{id}	t_{id}	$\log_{10}\mathsf{FAR}$	$\log_{10}\mathsf{FRR}$	Silicon area (μm^2)
SRAM PUF	143	34	−9.11	−9.04	116.2
Latch PUF	14881	4243	−9.00	−9.02	123523.9
D Flip-Flop PUF	703	201	−9.00	−9.09	8409.9
Buskeeper PUF	355	115	−9.02	−9.09	1646.7
Arbiter PUF (basic)	160	37	−9.04	−9.39	1089.8
Arbiter PUF (2-XOR)	226	67	−9.15	−9.09	2179.7
Ring Oscillator PUF (P.C.)	104	20	−9.83	−9.28	12238.3
Ring Oscillator PUF (L.G.)	192	48	−9.13	−9.36	3765.6

Table 5.3 Comparison of PUF parameters and estimated silicon area for an identification system with $\mathsf{EER} \leq 10^{-12}$

PUF class	n_{id}	t_{id}	$\log_{10}\mathsf{FAR}$	$\log_{10}\mathsf{FRR}$	Silicon area (μm^2)
SRAM PUF	196	47	−12.0	−12.1	159.3
Latch PUF	20464	5835	−12.0	−12.0	169867.2
D Flip-Flop PUF	968	277	−12.0	−12.1	11580.1
Buskeeper PUF	488	158	−12.1	−12.0	2263.7
Arbiter PUF (basic)	217	50	−12.1	−12.1	1089.8
Arbiter PUF (2-XOR)	312	93	−12.1	−12.3	2179.7
Ring Oscillator PUF (P.C.)	155	29	−14.8	−12.4	18828.1
Ring Oscillator PUF (L.G.)	260	65	−12.1	−12.2	5648.4

their very small cell area. Pairwise comparison ring oscillator PUFs, though showing the overall best identification performance in Fig. 5.3, are very area-inefficient, which makes them almost the least favorable choice for a required performance. Latch PUFs still behave the worst, even when taking into account their silicon area, because they require huge response lengths to provide meaningful levels of identification performance.

5.3 PUF-Based Entity Authentication

5.3.1 Background: PUF Challenge-Response Authentication

When an entity wants to *authenticate* itself to an another party, typically called the *verifier*, it needs to provide, besides plain identification, also corroborative evidence of its presented identity, i.e. evidence which could only have been created by that particular entity. An entity typically achieves this goal by proving to the verifier that it knows, possesses or contains a particular secret which only that entity can know, have or contain. In addition, the entity also needs to convince the verifier that it was actively, i.e. at the time of authenticating, involved in creating that evidence. In this section, we discuss how an entity can authenticate itself based on the possession/containment of a unique PUF instance. An authentication scenario is considered with a centralized verifier and entities which authenticate to the central verifier, not to each other.

There are two main approaches towards developing a PUF-based authentication system. The first approach is to develop an authentication scheme which directly deploys the unique and unpredictable challenge-response behavior of a particular PUF instance. The second approach consists of deriving a robust and secure cryptographic key from a PUF response and using this key in an existing classic key-based cryptographic authentication protocol. PUF-based key generation is discussed in detail in Chap. 6. In this section, we focus on the challenge-response approach, which is usually more efficient, since it does not require an additional implementation of a key generation algorithm and other keyed cryptographic primitives.

Basic PUF-Based Challenge-Response Authentication

The basic PUF-based challenge-response entity authentication scheme, among others described by Gassend et al. [42], Ranasinghe et al. [108] and Devadas et al. [30], consists of two phases, *enrollment* and *verification*:

1. Before deployment, every entity goes through enrollment by the verifier. During the enrollment phase, the verifier records the identity[3] ID of every entity, and

[3] Identification can happen with both assigned or inherent identifiers, as discussed in detail in Sect. 5.2. For simplicity, but without loss of generality, we assume an entity identifies with an assigned identifier ID in this section.

collects a significant subset of challenge-response pairs of every entity's PUF, for randomly generated challenges. The collected challenge-response pairs are stored in the verifier's database DB, indexed by the entity's ID.
2. During the verification phase, an entity identifies itself to the verifier by sending its ID. The verifier looks up the ID in its DB, and selects a random PUF challenge-response pair stored with that ID. The PUF challenge is sent to the entity; the entity evaluates its PUF with that challenge and replies with the obtained response. The verifier checks whether the replied response is *close to* the response it has in its database, i.e. both responses differ no more than some predetermined authentication threshold t_{auth}. If this check succeeds, the entity is authenticated, otherwise the authentication is rejected. The used challenge-response pair is removed from DB.

The correctness of this authentication scheme is ensured by the fact that PUF responses are reproducible over time, up to a small intra-distance. If t_{auth} is set large enough such that with high probability the intra-distance is smaller, the authentication of a legitimate entity succeeds. For the security of the authentication, the verifier relies on the fact that PUFs are unique and unclonable, and only the genuine PUF can with high probability reproduce a close response to a previously unobserved and random challenge.

Drawbacks of the Basic Protocol

Unfortunately, while strikingly simple and low-cost, this basic protocol exhibits a number of major shortcomings and drawbacks:

- It is clear that challenge-response pairs cannot be reused in order to avoid replay attacks, and a used pair is therefore removed from DB after the protocol finishes. This entails that the verifier needs to store a large number of pairs for each entity to make sure that every entity can be authenticated a reasonable number of times. Maintaining such a large database is a significant effort. Moreover, it requires that the considered PUF construction has a large challenge set to begin with, which already rules out a significant number of proposed PUF constructions.
- When the stored challenge-response pairs in DB of an entity run out, the entity can no longer be authenticated by the verifier. To make further authentications possible, the entity needs to be re-enrolled. This requires either a physical withdrawal of the entity from the field to undergo re-enrollment at the verifier's secured premises, or an elaborate and costly extension of the basic protocol to make remote re-enrollment possible.
- The basic protocol only provides authentication of an entity to the verifier; no mutual authentication can be supported without significantly extending the basic protocol.
- The basic protocol is only secure for truly unclonable PUFs (cf. Definition 13), i.e. PUFs which are also mathematically unclonable. PUFs which can be cloned in a mathematical sense (and which are hence still considered PUFs according to

Definition 16) do not offer secure authentication, since the basic protocol can no longer distinguish between the real entity with the physical PUF and an impersonator with a mathematical clone of that PUF. From Table 3.1, it is clear that only the optical PUF presents convincing evidence to be considered mathematically unclonable, which means the basic scheme is currently only secure when an optical PUF is deployed.[4]

Improvements and extensions of this basic protocol have been proposed, e.g. by Bolotnyy and Robins [9] and Kulseng et al. [71], to relax the strict requirements on the PUF construction and the database. However, these proposals do not completely succeed in overcoming the shortcomings of the basic protocol, or they introduce new restrictions. Some of these proposals are moreover shown to be insecure, e.g. Kardas et al. [63] point out significant security weaknesses of the proposal by Kulseng et al. [71].

Motivation

The simplicity of the basic PUF-based challenge-response authentication protocol is very appealing, especially when entities are heavily constrained silicon devices such as RFID tags, which cannot dedicate many resources to implementations of cryptographic building blocks. A possibly significant improvement in the resource-security trade-off of such devices is possible, if they can be authenticated only based on an efficient embedded intrinsic PUF implementation. Unfortunately, no intrinsic PUF candidates currently exist which meet the very strong requirement of mathematical unclonability needed to make the basic challenge-response authentication protocol secure. Constructing a practical intrinsic PUF for which strong guarantees of mathematical unclonability can be provided is currently considered an important open problem in the study of PUFs and their constructions.

Instead of attempting to tackle this difficult open problem, we will propose an alternative PUF-based authentication protocol in this section, which has considerably relaxed requirements for the PUF used, and which moreover provides mutual authentication. We need a little more complexity than the basic protocol (less is virtually impossible), but aim to keep it at a minimum, especially on the entity's side, which we consider to be a resource-constrained device like an RFID tag.

5.3.2 A PUF-Based Mutual Authentication Scheme

Rationale

The premise on which we base our proposed protocol is that the amount of unpredictability in the challenge-response behavior of an intrinsic PUF instance is strictly

[4]Intrinsic PUF constructions have been proposed which are candidates for mathematical unclonability, but currently they lack convincing argumentation to be classified as such.

limited, and increasing it comes at a high relative implementation cost. Note that this is the case for all currently known intrinsic PUF constructions. It is therefore not recommendable to publicly disclose challenge-response pairs in the protocol's communications, as the basic protocol does, since every disclosed pair significantly reduces the remaining unpredictability of the PUF's behavior. This quickly results in a situation where the PUF-carrying entity can no longer be securely authenticated.

In classic cryptographic challenge-response authentication protocols based on symmetric-key techniques (cf. [96, Sect. 10.3.2]), an entity also does not authenticate itself by disclosing its secret key, since this would render the authentication insecure after one protocol run. Instead, an entity only proves its knowledge of the secret key by showing that it can calculate encryptions or keyed one-way function evaluations of randomly applied challenges, without revealing any information about the key's value. We follow the same line of thought in our protocol: an entity demonstrates its possession of a PUF instance by demonstrating that it can calculate a function evaluation which takes an unpredictable PUF response as input, without fully disclosing the unpredictable nature of the response in the result of this evaluation.

An obvious choice for such a concealing function would be a one-way function, e.g. instantiated as a cryptographic hash function, since a one-way function evaluation of a PUF response does not disclose any significant information about the response value. The verifier could then calculate the same one-way function evaluation on the response in its database and check if this matches the entity's response. However, this runs into the problem of the non-perfect reproducibility of the PUF's responses. Whereas the entity's PUF response value will typically be *close to* the enrolled PUF response value in the verifier's database, the one-way function evaluations of both are completely independent and cannot be meaningfully matched to each other. This is also a consequence (in this case negative) of the one-way function destroying any predictability between input and output.

To overcome this issue, we need to deploy some form of error-correction in the protocol to make sure that both parties, the entity and the verifier, compute the one-way function on exactly the same PUF response value. In order to be able to do this, the entity and the verifier need to publicly exchange an amount of information about the PUF response. This is typically called *side information*, or also *helper data* in the context of PUF-based key generation algorithms (cf. Chap. 6). We show that it is possible for some intrinsic PUFs to find a good balance between the amount of side information one needs to disclose and the amount of unpredictability that is left in the PUF response after observing the side information, to obtain a reliable but secure authentication. Note that at no point in our proposed protocol do we derive a cryptographically secure key or do we use a keyed cryptographic primitive. Moreover, while being related to the key-generation algorithms discussed in Chap. 6, the used error-correction technique in this protocol is considerably less computationally expensive, especially on the entity's side.

Background The mutual authentication scheme as described in [151], of which a slightly adapted version is presented in this section, is the shared result of numerous fruitful discussions and an intense research collaboration between the author and Anthony van Herrewege, Roel Peeters and Prof. Ingrid Verbauwhede (all of the University of Leuven), and Christian Wachsmann, Prof. Stefan Katzenbeisser and Prof. Ahmad-Reza Sadeghi (all of the Technische Universität Darmstadt). The key idea and development of the "reverse" secure sketch is due to the author.

"Reversed" Secure Sketching Based on Linear Block Code Syndromes

The error-correction technique we deploy in the protocol is a practical version of the syndrome construction of a secure sketch, as described by Dodis et al. [32, 33], with the sketching procedure executed by the entity, and the recovery procedure executed by the verifier. Note that this execution order implies that the entity generates the "correct" version of the PUF response and the verifier needs to correct his stored response value to match that of the entity. This is the opposite way, as secure sketching is typically deployed, e.g. in PUF-based key generation, with the verifier storing a single fixed secret key and helper data string and requiring the entity to correct its noisy response, with the help of the helper data, to generate the same key as the verifier. For this reason, we call this a *reverse secure sketch*. The use of such a reverse secure sketch has some peculiar and interesting side effects:

- Since the entity's PUF response is only non-perfectly reproducible, the actual response value on which the authentication is based will possibly be different for each run of the protocol. This means the exchanged side information will also be different for each run. Since the side information partially discloses the response value, we need to consider this when showing the security of the protocol, i.e. we do not want multiple runs of the protocol to disclose the full response value. We are able to prove that even after an indeterminate number of protocol runs, with each a possibly different side information string based on a different noisy response value, the remaining unpredictability of the PUF response remains high.
- Since the sketching procedure of a secure sketch is typically much less computationally complex than the recovery procedure, it can be efficiently implemented by each entity using a small amount of resources. The central verifier is assumed to be less resource-constrained and easily capable of implementing the recovery procedure.

Secure sketch constructions based on linear block codes are explained in detail in Sect. 6.2.1. We refer to that section for more background on the operation of the syndrome construction used in our protocol. Summarized very briefly:

- The sketching procedure of the syndrome construction, executed by an entity, consists of a binary matrix multiplication of the entity's PUF response evaluation with the parity-check matrix of a linear block code, resulting in the side information for the protocol run: $w_i := y'_i \cdot \mathbf{H}^\mathsf{T}$.
- The recovery procedure, as executed by the verifier, consists of three steps: (i) a syndrome is calculated based on the received side information and the stored PUF response value from the verifier's database: $s_i := w_i \oplus y_i \cdot \mathbf{H}^\mathsf{T} \equiv e_i \cdot \mathbf{H}^\mathsf{T}$, with

Entity Ent_i : $\mathsf{ID}_i, \mathsf{puf}_i$		Verifier Ver : DB
$\mathsf{puf}_i \rightarrow y'_i$		
$w_i := y'_i \cdot \mathbf{H}^{\mathsf{T}}$		
$r_1 \overset{\$}{\leftarrow} \{0,1\}^{\ell}$	$\xrightarrow{\mathsf{ID}_i, w_i, r_1}$	Identify ID_i: $\mathsf{DB}[\mathsf{ID}_i] \rightarrow y_i$
		$y''_i := \mathsf{Recover}(y_i, w_i)$
		$r_2 \overset{\$}{\leftarrow} \{0,1\}^{\ell}$
$\mathsf{Hash}(\mathsf{ID}_i, w_i, y'_i, r_1, r_2) \overset{?}{=} u_1$	$\xleftarrow{u_1, r_2}$	$u_1 := \mathsf{Hash}(\mathsf{ID}_i, w_i, y''_i, r_1, r_2)$
· No match → Abort		
· Match → Accept Ver		
$u_2 := \mathsf{Hash}(\mathsf{ID}_i, y'_i, r_2)$	$\xrightarrow{u_2}$	$\mathsf{Hash}(\mathsf{ID}_i, y''_i, r_2) \overset{?}{=} u_2$
		· No match → Abort
		· Match → Accept Ent_i

Fig. 5.4 A PUF-based mutual authentication scheme between a PUF-carrying entity and a central verifier

$e_i = (y'_i \oplus y_i)$; (ii) the syndrome is decoded to an error string: $\mathsf{Decode}(s_i) \rightarrow e'_i$, with $e'_i = e_i$ if $\mathbf{HD}(y_i; y'_i) \leq t_{\mathsf{auth}}$, with t_{auth} the bit error correction capacity of the underlying code; (iii) the response reconstruction: $y''_i := y_i \oplus e'_i$, with $y''_i = y'_i$ if $\mathbf{HD}(y_i; y'_i) \leq t_{\mathsf{auth}}$.

The Mutual Authentication Protocol

The execution flow of our proposed PUF-based mutual authentication protocol is shown in Fig. 5.4. This is a modified version of the protocol we have introduced in [151], the main difference being the reversal of the authentication checks such that an entity will only authenticate to a legitimate verifier. We explain the different operations in more detail:

- Each entity (Ent_i) is assigned a unique identifier ID and equipped with a unique PUF instance puf_i. We only require a single challenge-response pair per PUF; hence we do not explicitly write the challenge: $\mathsf{puf}_i \rightarrow y'_i$.
- Prior to deployment, all entities are enrolled by the verifier (Ver), which keeps a database DB of a single response evaluation y_i of every entity's PUF, indexed by the entity's ID. PUF responses can only be enrolled once. This is enforced by physically blocking or even destroying the entity's enrollment interface which directly outputs the PUF response.
- Each entity implements the sketching procedure of the secure sketch, which is simply a binary matrix multiplication with a parity-check matrix $\mathbf{H}^{\mathsf{T}}$ of a linear block code. Due to the special structure of these matrices for many block codes, this multiplication can often be very efficiently implemented in digital hardware.
- The verifier implements the recovery procedure of the secure sketch: $\mathsf{Recover}(y_i, w_i)$. This procedure contains an error-correction decoding algorithm which typically requires a considerable computational effort.

- Each entity, as well as the verifier, implements a cryptographically secure hash function: $\mathsf{Hash}(\cdot)$.
- Each entity, as well as the verifier, has access to a cryptographically secure random bit generator which they use to generate random nonces used in the protocol: $r_i \leftarrow \{0,1\}^{\ell}$. The nonces are used to introduce freshness in the protocol's messages in order to avoid replay attacks. Later we propose a possible optimization in which entities do not need to produce a nonce and hence do not require a random bit generator.

Correctness

The correctness of the protocol is guaranteed by the error-correction capability of the underlying linear block code of the secure sketch. If, with high probability, the intra-distance between the PUF response produced by a legitimate entity during a protocol run and the response in the legitimate verifier's database is smaller than or equal to the error-correcting capacity t_{auth} of the underlying block code, then the verifier is able to recover the same response value produced by the entity. In that case, both hash value checks will succeed and mutual authentication is accomplished. Based on the intra-distance distribution of the deployed PUF construction, a code with an appropriate error-correction threshold is selected. This will ultimately result in a false rejection rate, in a manner equivalent to PUF-based identification as discussed in Sect. 5.2.2.

Security

Next, we discuss the different security aspects of the protocol. We refer to [151] for a detailed security analysis, including a security proof, of a variant of the presented protocol.

Physical Unclonability Due to the physical unclonability of the deployed PUFs, an impersonation attempt of an entity carrying a different PUF will fail with high probability. Equivalently to the identification system discussed in Sect. 5.2.2, a false acceptance rate can be computed which depends on the inter-distance distribution of the PUFs, the number of bits n_{auth} in the considered PUF responses, and the selected error-correction threshold t_{auth}. Note that this false acceptance rate expresses the probability that two different PUFs are, coincidentally, similar enough to impersonate each other. The actual false acceptance rate of the overall protocol also depends on the collision resistance of the used hash function and could be considerably higher.

Replay Attacks The random nonces r_1 and r_2 respectively generated by the entity and the verifier preclude replay attacks from both sides, by introducing freshness in the protocol communications. An adversary trying to replay protocol messages in order to impersonate the verifier will fail since the entity will present a different nonce value and the adversary cannot recompute the hash evaluation over this new nonce since he doesn't know the PUF response. The same holds for an adversary trying to impersonate an entity by replaying earlier recorded messages from a successful entity authentication. Under certain conditions, the nonce generated by the entity can even be omitted. This is the case if the expected intra-distance on the entity's PUF response is *large* enough such that it is highly unlikely that exactly the same response value will ever be reproduced. In that case, the freshness of the protocol is guaranteed by the *noise* on the entity response, given that it is large enough. A practical advantage of this variant is that entities do not need access to a random bit generator any longer, which reduces the resource requirements of the protocol.

Response Unpredictability We still need to assess the unpredictability of responses, given that the adversary can observe multiple side information strings from many successful authentication attempts of an entity. We first consider the unpredictability of a response after observing one side information string. After observing a protocol run, an adversary can launch an offline attack on the unknown response value y'_i based on the observed quartet $(\mathsf{ID}_i, r_1, w_i, u_1)$, by guessing a response value y^*_i and checking whether $\mathsf{Hash}(\mathsf{ID}_i, w_i, y^*_i, r_1) \overset{?}{=} u_1$. Note that, by observing the side information, an adversary gains quite some information about the response value y'_i, since every bit of w_i is a linear combination of bits from y'_i, which helps it in the guessing attack. We express the remaining unpredictability of the response as its conditional entropy when conditioned on the side information. It is rather trivial to show that $H(Y'_i|W_i) \geq H(Y'_i) - |W_i|$. This means that to ensure there is any unpredictability left, the length of the side information needs to be smaller than the response's entropy. Now we show that the unpredictability of the PUF response remains this high even after observing many side information strings computed over different (possibly noisy) evaluations of the same response, by proving the following lemma:[5]

Lemma 1 *If* $\forall j$: Y'_i *and* D^{intra}_j *are pairwise independently distributed, then*:

$$H\big(Y'_i|f\big(Y'_i\big), f\big(Y'_i \oplus D^{\mathsf{intra}}_1\big), f\big(Y'_i \oplus D^{\mathsf{intra}}_2\big), \dots, f\big(Y'_i \oplus D^{\mathsf{intra}}_q\big)\big) = H\big(Y'_i|f\big(Y'_i\big)\big),$$

for any positive integer q *and for any linear function* f.

Proof

$$H\big(Y'_i|f\big(Y'_i\big), f\big(Y'_i \oplus D^{\mathsf{intra}}_1\big), f\big(Y'_i \oplus D^{\mathsf{intra}}_2\big), \dots, f\big(Y'_i \oplus D^{\mathsf{intra}}_q\big)\big),$$

(f is a linear function),

$$= H\big(Y'_i|f\big(Y'_i\big), f\big(Y'_i\big) \oplus f\big(D^{\mathsf{intra}}_1\big), f\big(Y'_i\big) \oplus f\big(D^{\mathsf{intra}}_2\big), \dots, f\big(Y'_i\big) \oplus f\big(D^{\mathsf{intra}}_q\big)\big),$$

[5]Note that this proof differs from the security proof given in [151].

$$= H\big(Y_i' | f(Y_i'), f(D_1^{\mathsf{intra}}), f(D_2^{\mathsf{intra}}), \ldots, f(D_q^{\mathsf{intra}})\big),$$
$$\big(\forall j : Y_i' \text{ and } D_j^{\mathsf{intra}} \text{ are independent}\big),$$
$$= H\big(Y_i' | f(Y_i')\big). \qquad \square$$

The assumption of independence between the distributions of response values and intra-distances is very reasonable since they originate from different physical processes. This lemma is directly applicable to our situation, since every helper data string is a linear function of a response evaluation. Based on a similar reasoning, one can also show that the remaining *min-entropy* remains high. This is proven in a generalized form by Boyen [13]. The response's unpredictability as expressed by $H(Y_i'|W_i)$ measures the resistance against an offline attack; hence it needs to be sufficiently large to offer long-term security.

5.3.3 Authentication Performance of Different Intrinsic PUFs

Proposed Entity Design

We first determine a concrete and realistic design of an entity which is used to compare the authentication performance of different intrinsic PUFs in the proposed protocol. The main design choice is the selection of the underlying error-correcting linear block code of the secure sketch. We refer to Burr [18, Chap. 6] for a detailed introduction to block codes. We propose using a concatenation of a simple repetition code followed by a BCH code as introduced by Hocquenghem [51] and Bose and Ray-Chaudhuri [12], which yields an overall syndrome construction with a relatively high error correction performance. The BCH code's parameters are $[n_{\mathsf{BCH}}, k_{\mathsf{BCH}}, t_{\mathsf{BCH}}]$, which means that it has code words of length n_{BCH} and dimension k_{BCH}, and up to t_{BCH} bit errors in a single code word can be corrected. The corresponding parameters of the repetition code are $[n_{\mathsf{REP}}, 1, \frac{n_{\mathsf{REP}}-1}{2}]$, with n_{REP} odd.

An $[n, k, t]$ binary linear block code can be fully described by its generation matrix $\mathbf{G_{k\times n}}$ or its corresponding parity-check matrix $\mathbf{H_{(n-k)\times n}}$, which meet the condition $\mathbf{G} \cdot \mathbf{H}^{\mathsf{T}} = \mathbf{0}$. For a repetition code, a multiplication of an n_{REP}-bit word with the code's parity-check matrix can be implemented straightforwardly in hardware using $(n_{\mathsf{REP}} - 1)$ 2-input XOR gates. Due to its special algebraic structure, the multiplication of an n_{BCH}-bit word with the parity-check matrix of a BCH code can also be implemented very efficiently using an $(n_{\mathsf{BCH}} - k_{\mathsf{BCH}})$-bit linear feedback shift register (LFSR) with the feedback taps determined by the generator polynomial of the BCH code. The overall side information generation of a concatenated repetition and BCH code can hence be efficiently implemented using only minimal resources on the entity's side. A schematic representation of the side information generator is shown in Fig. 5.5. Note that in the proposed design, the BCH code's side information and the verifier authentication hash value are only computed over

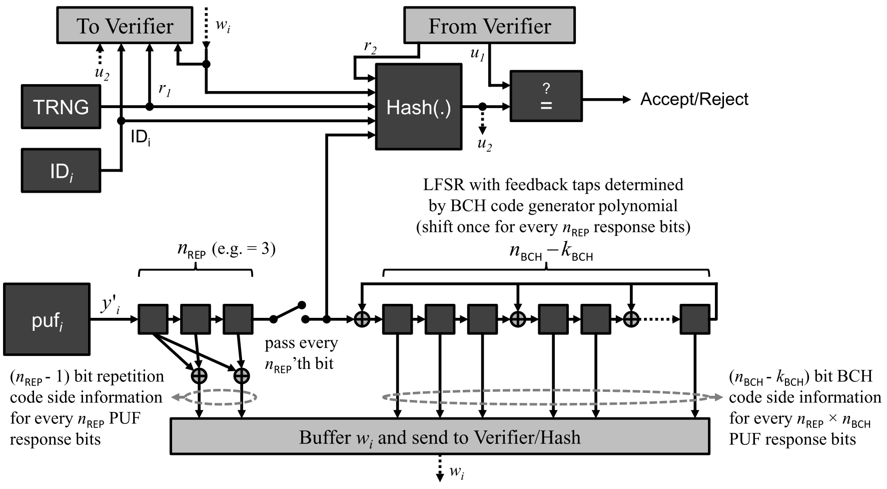

To Verifier
w_i
From Verifier
r_2
u_1
u_2
r_1
TRNG
Hash(.)
?
=
Accept/Reject
ID_i
ID_i
u_2
LFSR with feedback taps determined by BCH code generator polynomial (shift once for every n_{REP} response bits)
n_{REP} (e.g. = 3)
$n_{\mathrm{BCH}} - k_{\mathrm{BCH}}$
puf_i
y'_i
pass every n_{REP}'th bit
(n_{REP} - 1) bit repetition code side information for every n_{REP} PUF response bits
(n_{BCH} - k_{BCH}) bit BCH code side information for every $n_{\mathrm{REP}} \times n_{\mathrm{BCH}}$ PUF response bits
Buffer w_i and send to Verifier/Hash
w_i

Fig. 5.5 Design of an entity's side information generator, based on a concatenated repetition and BCH block code, and the corresponding verifier authentication check

every n_{REP}'th response bit. Since the remaining $n_{\mathsf{REP}} - 1$ bits are immediately disclosed by the repetition code's side information, it makes no sense to consider them further for authentication and they are discarded.

For the hash function implementation, we propose using a lightweight hash function. In our prototype implementation in [151] we used the SPONGENT hash function as proposed by Bogdanov et al. [8]. To keep the side information generator small, we constrain the repetition code to $n_{\mathsf{REP}} \leq 11$ and the BCH code to $n_{\mathsf{BCH}} \leq 255$. We now investigate the implementation parameters of this proposed design, within these constraints, when the deployed PUF is one of the intrinsic PUFs studied in Chap. 4. In particular, we determine the required code parameters based on the intrinsic PUFs' statistics summarized in Table 4.10. Based on these code parameters, the required number of PUF response bits, and ultimately the required PUF silicon area to reach a particular authentication performance, can be calculated.

Example of Authentication Performance Calculation

We demonstrate how the authentication performance is calculated for a single PUF in an exemplary but realistic scenario. Next, we present the results of the same calculations for all the intrinsic PUFs studied in a number of possible scenarios.

As an example, we consider the SRAM PUF, and we aim for an authentication performance with $\mathsf{EER} \leq 10^{-9}$ and an unpredictability of at least 128 bits, i.e. $H(Y|W) \geq 128$. Searching the design space constrained by the proposed entity design above yields the following parameters:

- A $[n_{\mathsf{REP}} = 3, 1, 1]$ repetition code and a $[n_{\mathsf{BCH}} = 223, k_{\mathsf{BCH}} = 83, t_{\mathsf{BCH}} = 21]$ BCH code are used. In total, $3\times$ a BCH code length is required.

- These code parameters achieve an overall $\mathsf{FRR} = 10^{-9.63}$ and $\mathsf{FAR} = 10^{-104.84}$.
- The total number of required PUF response bits is $n_{\mathsf{auth}} = 3 \times n_{\mathsf{REP}} \times n_{\mathsf{BCH}} = 2007$.
- The total number of exchanged side information bits is $\ell_{\mathsf{auth}} = 3 \times (n_{\mathsf{REP}} \times n_{\mathsf{BCH}} - k_{\mathsf{BCH}}) = 1785$.
- The remaining entropy is calculated as $H(Y|W) \geq H(Y) - |W| = n_{\mathsf{auth}} \times \rho(Y^{n_{\mathsf{auth}}}) - \ell_{\mathsf{auth}} = 2007 \times 94.09\,\% - 1758 = 130.4$ bits.

Note that these parameters are optimized within the given constraints to yield to smallest possible number of required PUF bits. For most intrinsic PUFs, the entropy density $\rho(Y^{n_{\mathsf{auth}}})$ is a constant and independent of the number of considered response bits, as shown in Table 4.10. However, for both arbiter-based PUFs, $\rho(Y^{n_{\mathsf{auth}}})$ is a decreasing function of the required number of bits n_{auth}, of which an upper bound is given by Fig. 4.5.

Performance Comparison of Different Intrinsic PUFs

Now we apply the same calculation as in this example on all studied intrinsic PUFs for three different authentication performances:

1. A low-cost security scenario with $\mathsf{EER} \leq 10^{-6}$ and $H(Y|W) \geq 80$.
2. A realistic security scenario with $\mathsf{EER} \leq 10^{-9}$ and $H(Y|W) \geq 128$.
3. A critical security scenario with $\mathsf{EER} \leq 10^{-12}$ and $H(Y|W) \geq 256$.

Based on the number of response bits needed, the required silicon area is also calculated in the same manner (and with the same disclaimers) as in Sect. 5.2.3. The results are presented in Table 5.4.

From Table 5.4, it is clear that for many studied intrinsic PUFs, no parameter solutions meeting the presented design constraints can be found. This is mostly a result of a too high average intra-distance, a too low entropy density, or a combination of both. Only the SRAM PUF and the pairwise comparison ring oscillator PUF succeed in providing a solution for all three considered scenarios. The SRAM PUF solution offers a better area efficiency than the ring oscillator PUF by almost two orders of magnitude.

5.4 Conclusion

In Sect. 5.2 of this chapter, we have successfully demonstrated that a PUF response can be used as a secure and reliable inherent identifier of a PUF-embedding entity. The identification performance, in terms of false acceptance, false rejection and equal error rate, scales only with the bit length of the PUF response. For the intrinsic PUFs studied in Chap. 4, this ultimately comes down to a scaling with the required silicon area needed to implement the PUF. The scaling factor is different for every intrinsic PUF construction and is determined by the characteristics of its inter-

Table 5.4 Required code parameters, PUF response bits and estimated silicon area for achieving three different authentication performances. '/' means that no parameter solution within the given design constraints can be found

PUF class	$\mathsf{EER} \leq 10^{-6}$ and $H(Y'\|W) \geq 80$								Silicon area (μm^2)
	$\times$BCH	n_{REP}	n_{BCH}	k_{BCH}	t_{BCH}	$\log_{10}\mathsf{EER}$	$H(Y'\|W)$	n_{auth}	
SRAM PUF	1	3	248	124	18	−7.05	80	744	604.5
Latch PUF	/	/	/	/	/	/	/	/	/
D Flip-Flop PUF	/	/	/	/	/	/	/	/	/
Buskeeper PUF	/	/	/	/	/	/	/	/	/
Arbiter PUF (basic)	/	/	/	/	/	/	/	/	/
Arbiter PUF (2-XOR)	1	5	232	100	19	−6.2	80	1160	2179.7
Ring Oscillator PUF (P.C.)	2	1	216	52	25	−6.1	81	432	50835.9
Ring Oscillator PUF (L.G.)	/	/	/	/	/	/	/	/	/
	$\mathsf{EER} \leq 10^{-9}$ and $H(Y'\|W) \geq 128$								
SRAM PUF	3	3	223	83	21	−9.63	130	2007	1630.7
Arbiter PUF (2-XOR)	/	/	/	/	/	/	/	/	/
Ring Oscillator PUF (P.C.)	2	3	167	91	10	−9.19	128	1002	117911.1
	$\mathsf{EER} \leq 10^{-12}$ and $H(Y'\|W) \geq 256$								
SRAM PUF	7	3	249	81	26	−12.37	257	5229	4248.7
Ring Oscillator PUF (P.C.)	8	1	254	46	42	−14.35	260	2032	239117.2

and intra-distance distributions. A comparative analysis, as presented in Tables 5.2 and 5.3, indicates that for a given silicon area, the SRAM PUF provides the best identification performance in terms of equal error rate. The latch PUF on the other hand exhibits very poor identification capabilities, which severely undermines its usage as a PUF.

We build upon these identification capabilities of intrinsic PUFs to provide secure entity authentication. In order to obtain a cryptographic level of authentication security, we proposed a new authentication protocol between a central verifier and a deployed entity, shown in Fig. 5.4, that utilizes a unique and unpredictable response of a PUF instance embedded by each entity as an authentication secret. The protocol requires some form of error-correction which we accomplish by using a secure sketch in a 'reversed' mode of operation, i.e. an entity generates new side information in each protocol run and the verifier uses this side information to modify its fixed PUF response to match the current response evaluation of the entity. Based on the linearity of the sketching procedure, we are able to prove that even after many protocol runs, the unpredictability of the PUF response remains high. The authentication performance metrics of this protocol are derived in a similar way as for the PUF-based identification, and applied on the experimental intrinsic PUF results. From this analysis, summarized in Table 5.4, it is clear that most of the intrinsic PUFs studied in Chap. 4 are not able to achieve a high-level authentication performance within a resource-constrained environment. The SRAM PUF and the pairwise-comparison ring oscillator PUF, and to a lesser extent also the 2-XOR arbiter PUF, are the only constructions which are sufficiently unpredictable and reproducible to be practically usable in the protocol, with the SRAM PUF exhibiting the best silicon area efficiency.