Chapter 3
Physically Unclonable Functions: Properties

3.1 Introduction

3.1.1 Motivation

In Chap. 2, we introduced the PUF concept and illustrated it by means of an extensive enumeration of exemplary constructions which have been proposed over the years. From this list it is clear that the term 'PUF' has been used, in printed publications but even more so in colloquial speech, as a label for a wide variety of different constructions. However, intuitively it is clear that all these constructions share a number of specific properties. When properly and unambiguously described, these properties allow us to *define* a PUF, i.e. to identify the specific attributes which make us intuitively label certain constructions as PUFs and others not.

In many publications which introduce a new PUF construction, a definition for the more general PUF concept is attempted, using varying degrees of formalism. While often fitting for the construction proposed in the same publication, most of these definitions run into problems when applied to the wider group of different PUF constructions, as discussed in Chap. 2:

- Some proposed definitions are too strict since they clearly exclude a number of constructions which are labelled as PUFs, e.g. the early definition of a physical one-way function as proposed by Pappu [104] includes a one-wayness property, but almost none of the PUF constructions proposed afterwards meet this very strict condition.
- Other definitions are too loose, i.e. they apply equivalently to constructions which are generally not considered as PUFs, e.g. true random number generators.
- Many of the proposed definitions are an ad hoc listing of the perceived qualities of the simultaneously proposed new PUF construction, and lack a degree of generalization.

These issues, combined with the fact that there are nearly as many different PUF definitions as there are PUF constructions, have caused confusion and even lead to problematic situations. One particular problem which occurs regularly is that some of these ad hoc properties which often appear in PUF definitions are quickly generalized to *all* PUFs, whereas many proposed PUF constructions do in fact not meet them. It becomes even worse when some recurring properties are in practice even *assumed* for a newly proposed PUF construction, without ever actually verifying if the construction meets them. It goes without saying that this can lead to disastrous failures when security-sensitive systems using such a PUF construction rely on these properties.

Most of these PUF definitions are moreover of an informal nature. While not problematic by itself, this does cause issues when PUFs are deployed in formal systems such as cryptographic algorithms and protocols. Designers of such formal systems are typically not familiar with the physical intricacies of a PUF construction and need to rely on a strictly defined formal model of the PUF. On the other hand, it turns out to be particularly difficult to capture the physical behavior in an unambiguous and meaningful formal description. Either the description is not formal enough, which significantly reduces the usability for a formal designer, or it is too formal, which makes it impossible for practice-oriented PUF developers to evaluate the strict formal conditions for their PUF construction. Especially in the latter case, there is the risk of introducing a model which is picked up in the formal world, but which has no realistic connection to actual implementations any more.

3.1.2 Chapter Goals

Whereas Chap. 2 was intended as an exploration of the expanding field of physically unclonable functions, the goal of this chapter is to introduce a classification in this large and widely differing collection of PUF proposals based on the algorithmic properties of their challenge-response behavior. In this chapter we aim to:

- Give a detailed overview of significant properties which have, at one point or another, been attributed to PUFs, and propose a semi-formal definition of these properties based on their intuitive description.
- Make a comparative analysis of the defined properties on a representative subset of PUF proposals in order to distinguish between defining and nice-to-have characteristics. This analysis will yield a much more tactile definition of what we have intuitively called a PUF in Chap. 2.
- Discuss a formal framework for deploying PUFs in formal security models, which is partially based on this semi-formal study. The objective is to develop a framework which is formal enough to allow meaningful security reductions, but at the same time sufficiently realistic and flexible to actually demonstrate the formally defined properties for real PUF implementations.

3.1.3 Chapter Overview

In Sect. 3.2, we do a study on the many different PUF properties which have been proposed over time. By means of a comparative analysis over a representative set of PUF constructions, we identify which properties are defining and which are only nice to have. This study is an extended and updated version of our earlier work in [83, Sect. 4]. In Sect. 3.3, we propose a set of formal definitions of the most important PUF properties, which is intended as a carefully balanced interface between practical PUF developers and theorists. This section is based on our work in [3]. We conclude this chapter in Sect. 3.4.

3.2 A Discussion on the Properties of PUFs

In this section, we start by listing a number of properties which are sensible to assess for PUFs, and most of them have therefore been attributed at one or more occasions to particular PUF constructions. We define these properties in a semi-formal way using the notation introduced in Sect. 2.2. The definitions are semi-formal in the sense that, while attempting to be as unambiguous as possible, we refrain from introducing a plethora of quality parameters which would make the notation needlessly complex. In that respect, we use informal qualifiers like *easy* and *hard*, *small* and *large*, and *high* and *low* to express the bounds imposed by most properties.

To avoid confusion, we want to state explicitly that at this moment we only define a number of properties, and we make no claims yet as to which PUF constructions satisfy which properties, or as to which properties are naturally implied for all PUFs. Such an analysis is only done in Sect. 3.2.9 based on a representative subset of proposed PUF constructions and is discussed in Sect. 3.2.10.

3.2.1 Constructibility and Evaluability

The notion of evaluability of a PUF construction is a purely practical and rather basic consideration expressing the fact that the required effort to obtain a meaningful outcome of a PUF instance should be feasible. Before defining evaluability, we first want to introduce the even more basic notion of constructibility, i.e. the condition that it is actually possible to produce instantiations of a particular PUF design.

Constructibility

Definition 5 A PUF class $\mathcal{P}$ is constructible if it is easy to invoke its Create procedure and produce a random PUF instance $\mathsf{puf} \leftarrow \mathcal{P}.\mathsf{Create}(r^{\mathsf{C}} \overset{\$}{\leftarrow} \{0,1\}^*)$.

It is hard to discuss the remaining properties for proposals which have no feasible instantiations. Constructibility is therefore a conditio sine qua non for evaluability, and by extension for all of the following properties listed here. The qualifier 'easy' in the definition is context-dependent. Since PUFs are physical objects, their constructibility requires at least that they be possible within the laws of physics. From a more practical viewpoint, 'easy' relates to the cost of producing an instance of a particular PUF class. An important detail in the definition of constructibility is that it is merely easy to construct a *random* PUF instance, i.e. without any specific requirements on its challenge-response behavior, whereas constructing a *specific* PUF instance can be infeasibly hard. In Sect. 3.2.4, we will discuss why it is even desirable that this is hard.

Evaluability

Definition 6 A PUF class $\mathcal{P}$ exhibits evaluability if it is constructible, and if for any random PUF instance $\mathsf{puf} \in \mathcal{P}$ and any random challenge $x \in \mathcal{X}_{\mathcal{P}}$ it is easy to evaluate a response $y \leftarrow \mathsf{puf}(x).\mathsf{Eval}(r^{\mathsf{E}} \overset{\$}{\leftarrow} \{0,1\}^*)$.

Since the following properties all deal with the challenge-response behavior of PUF instances, it is hard to discuss the meaningfulness of constructions which are not evaluable. The 'easiness' expressed in the definition is again context-dependent. In a theoretical treatise, this typically points to some variant of 'in polynomial time and effort'. Practically however, an easy evaluation means an evaluation which is possible within the strict timing, area, power, energy and cost budget imposed by the application. From this point of view, an evaluation which is easy for one application could be infeasible for another one.

3.2.2 *Reproducibility*

The first property of a PUF's challenge-response behavior we will discuss is its reproducibility, which is technically also of a practical nature. However, as will become clear later on, it also has strong repercussions on the attainable security parameters of PUF-based applications.

Definition 7 A PUF class $\mathcal{P}$ exhibits reproducibility if it is evaluable, and if

$$\Pr\big(D^{\mathsf{intra}}_{\mathcal{P}} \text{ is small}\big) \text{ is high.}$$

Reproducibility is defined with respect to the distribution of the response intra-distance of the entire PUF class, i.e. considering evaluations of random challenges on random PUF instances. This means that with high probability, responses resulting from evaluating the same challenge on the same PUF instance should be similar, i.e. close in the considered distance metric. If the evaluation condition (α) has an impact on the responses, the definition is extended by considering the response intra-distance under condition α: $D^{\mathsf{intra}}_{\mathcal{P};\alpha}$. The qualifiers 'small' and 'high' in the definition are context-specific. Whether an intra-distance is ‘small’ is generally determined in relation to similar notions in other properties such as uniqueness (cf. Definition 8), e.g. as made explicit in the definition of identifiability (cf. Definition 9). How ‘high’ the probability needs to be typically follows from the application requirements.

3.2.3 Uniqueness and Identifiability

The most basic security-related property of PUFs is uniqueness: the observation that a PUF response is a measurement of a random and instance-specific feature.

Uniqueness

Definition 8 A PUF class $\mathcal{P}$ exhibits uniqueness if it is evaluable, and if

$$\Pr\bigl(D_{\mathcal{P}}^{\mathsf{inter}} \text{ is large}\bigr) \text{ is high.}$$

In the same way as reproducibility, uniqueness is defined with respect to the distribution of the response inter-distance random variable of the entire PUF class, i.e. considering evaluations of random challenges on random pairs of PUF instances. This means that, with high probability, responses resulting from evaluating the same challenge on different PUF instances should be dissimilar, i.e. far apart in the considered distance metric. Uniqueness is generally assessed at nominal operating conditions; hence there is no need to extend this definition to varying evaluation conditions. The qualifiers ‘large’ and ‘high’ are again context-specific.

Identifiability

When a PUF class exhibits both reproducibility and uniqueness, it follows that its PUF instances can be identified based on their responses. We express this in the separate property of identifiability.

Definition 9 A PUF class $\mathcal{P}$ exhibits identifiability if it is reproducible and unique, and in particular if

$$\Pr\bigl(D_{\mathcal{P}}^{\mathsf{intra}} < D_{\mathcal{P}}^{\mathsf{inter}}\bigr) \text{ is high.}$$

Identifiability expresses the fact that responses (to the same challenge) coming from a single PUF instance are more alike than responses coming from different instances. This means that, using their response evaluations, instances of the PUF class can state a static identity which is with high probability unique. The details of how such an identification scheme can be implemented are given in Sect. 5.2. The extent to which a PUF class is identifiable is often quickly estimated based on experimental results, by comparing the average observed intra- and inter-distances and demonstrating that $\mu_{\mathcal{P}}^{\text{intra}} \ll \mu_{\mathcal{P}}^{\text{inter}}$. From this and the following definitions, it is also evident why the distributions of $D_{\mathcal{P}}^{\text{inter}}$ and $D_{\mathcal{P}}^{\text{intra}}$ play a pivotal role in the assessment of the usability of a particular PUF class, and why it is important to get accurate information about these distributions from experimental statistics.

3.2.4 Physical Unclonability

Assume an adversary which has control over the creation procedure of a PUF class, i.e. it can influence the conditions, parameters and randomness sources of $\mathcal{P}.\mathsf{Create}$ to a certain (feasible) extent. When considering identification based on PUF responses in the presence of such an adversary, a stronger argument is required to ensure that all PUF-based identities are unique with high probability. This is because this adversary can use its control over the instance creation process to attempt to produce two PUF instances which are more alike than one would expect based on the uniqueness property. To avoid this, one would like to *enforce* the uniqueness property, i.e. to ensure that the uniqueness property is met, even in the presence of such an adversary. This is what we call *physical unclonability*.

Definition 10 A PUF class $\mathcal{P}$ exhibits physical unclonability if it is evaluable, and if it is hard to apply and/or influence the creation procedure $\mathcal{P}.\mathsf{Create}$ in such a way as to produce two distinct PUF instances puf and $\mathsf{puf}' \in \mathcal{P}$ for which it holds that

$$\Pr\big(\mathbf{dist}\big[Y \leftarrow \mathsf{puf}(X); Y' \leftarrow \mathsf{puf}'(X)\big] < D_{\mathcal{P}}^{\text{inter}}(X)\big) \text{ is high,}$$

for $X \leftarrow \mathcal{X}_{\mathcal{P}}$. In extremis, it should be very hard to produce two PUF instances for which it holds that

$$\Pr\big(\mathbf{dist}\big[Y \leftarrow \mathsf{puf}(X); Y' \leftarrow \mathsf{puf}'(X)\big] > D_{\mathcal{P}}^{\text{intra}}(X)\big) \text{ is low.}$$

The qualifier 'hard' in this definition reflects the physical and technical difficulties (or impossibility) in creating such a PUF instance pair. These difficulties need to be evaluated with respect to the technical capabilities of the adversary, which ultimately is a function of its expertise and its equipment budget. Note that the difficulty of producing a non-unique PUF instance pair, as described in the definition, implies the difficulty of producing a single PUF instance which is more alike to a given PUF instance than expressed by the uniqueness property. When combined with constructibility, physical unclonability can be summarized as: *it is easy to create a random PUF instance, but hard to create a specific one.*

A PUF class which exhibits physical unclonability has the interesting security advantage that even the genuine manufacturer of PUF instances has no way of breaking the uniqueness property. This means that one does not need to *trust* the manufacturer to make sure every PUF instance is unique with high probability, since this is implied by the physical unclonability of the PUF class. This advantage of 'physically unclonable PUFs' is called *manufacturer resistance*.

3.2.5 Unpredictability

Many applications of PUFs rely on their challenge-response functionality, i.e. the ability to apply a challenge and to receive a random response in reply. In that respect, uniqueness, and by extension physical unclonability, are often not sufficient to ensure security. One also requires unpredictability between responses on a single PUF instance, i.e. unobserved responses remain sufficiently random, even after observing responses to other challenges on the same PUF instance.

Definition 11 A PUF class $\mathcal{P}$ exhibits unpredictability if it is evaluable, and if it is hard to win the following game for a random PUF instance $\mathsf{puf} \in \mathcal{P}$:

- In a learning phase, one is allowed to evaluate puf on a limited number of challenges and observe the responses. The set of evaluated challenges is $\mathcal{X}'_{\mathcal{P}}$ and the challenges are either randomly selected (weak unpredictability) or adaptively chosen (strong unpredictability).
- In a challenging phase, one is presented with a random challenge $X \leftarrow \mathcal{X}_{\mathcal{P}} \setminus \mathcal{X}'_{\mathcal{P}}$. One is required to make a prediction Y_{pred} for the response to this challenge when evaluated on puf. One does not have access to puf, but the prediction is made by an algorithm $\mathsf{predict}$ which is trained with the knowledge obtained in the learning phase: $Y_{\mathsf{pred}} \leftarrow \mathsf{predict}(X)$.
- The game is won if

$$\Pr\big(\mathbf{dist}\big[Y_{\mathsf{pred}} \leftarrow \mathsf{predict}(X);\, Y \leftarrow \mathsf{puf}(X)\big] < D^{\mathsf{inter}}_{\mathcal{P}}(X)\big) \text{ is high.}$$

Note the similarity in the expressions involving the distribution of the inter-distance in this definition and the definition of physical unclonability. However, instead of considering the distance to a second created PUF instance puf', here we consider the distance to a prediction algorithm $\mathsf{predict}$ which is trained on an earlier observed set of challenges and responses on the same PUF instance. We use a game-based description for unpredictability to avoid having to put any restrictions on the prediction algorithm, i.e. unpredictability is defined with respect to the best conceivable prediction algorithm which can be built, trained and evaluated within the capabilities of the adversary. In the best case, one can show that responses to different challenges are completely independent, which means they cannot be predicted by any prediction algorithm. However, such a strong quality can rarely be proven for a PUF construction, and at best one can assume it for certain PUFs based on a physical motivation.

For other PUF constructions, responses are not independent and their unpredictability relies on the computational difficulty of constructing, training and evaluating an appropriate prediction algorithm. In that case, the extent of unpredictability can only be estimated in relation to the currently best-known modeling attack, since there is no guarantee that no better attacks exist. This is similar to the situation for most cryptographic symmetric primitives, e.g. a block cipher like AES, where a primitive is only as secure as indicated by the currently best-known attack. Kerckhoffs' principle typically also applies to PUFs, i.e. an adversary has full knowledge about the design and implementation details of a particular PUF construction, except for the instance-specific random features introduced during the creation process. The efficiency of a practical model building attack aimed at breaking the unpredictability of a PUF is typically expressed by their prediction accuracy as a function of the size of the training set, e.g. as done in the description of modeling attacks on arbiter PUFs in Sect. 2.4.1.

3.2.6 Mathematical and True Unclonability

In the definition of unpredictability, an adversary is restricted to learning a limited number of (possibly random) challenge-response pairs which it uses to train its prediction algorithm. This is typically the case in a challenge-response-based protocol where the adversary eavesdrops on the protocol communications. However, a stronger adversarial model needs to be considered when the adversary has unlimited physical access to a PUF instance. This means it can learn as many challenge-response pairs as it is capable of storing and possibly even make useful observations of the PUF instance beyond the challenge-response functionality.

Definition 12 A PUF class $\mathcal{P}$ exhibits mathematical unclonability if it is unpredictable, even if there is no limit on the access to the PUF instance during the learning phase (as described in Definition 11), besides one's own capacities.

Mathematical unclonability is hence the extension of unpredictability to an adversary with unlimited physical access to a PUF instance. It is therefore evident that mathematical unclonability implies unpredictability.

A direct condition for a PUF class $\mathcal{P}$ to be mathematically unclonable is that its challenge set $\mathcal{X}_{\mathcal{P}}$ be very large, preferably exponential in some construction parameter of $\mathcal{P}$. If this is not the case, an adversary with unlimited physical access to a PUF instance can evaluate the complete challenge set and store the observed responses in a table. This table then serves as a perfect prediction model of the considered PUF instance and the unpredictability property is broken (technically, there are no challenges left to play the challenging phase of the unpredictability game with). The same argument holds if the challenge set is large, but if a near-perfect response prediction algorithm can be trained based on a small subset of these challenges.

True Unclonability

We have defined two different notions of unclonability: physical and mathematical unclonability. Both describe a property with the same objective, i.e. it is hard to *clone* a PUF instance, but from completely different perspectives. Physical unclonability deals with actual physical clones of PUF instances, whereas mathematical unclonability deals only with cloning the challenge-response behavior of a PUF instance. For a PUF class to exhibit true unclonability, both properties need to be met.

Definition 13 A PUF class $\mathcal{P}$ exhibits true unclonability if it is both physically and mathematically unclonable.

3.2.7 *One-Wayness*

We describe a one-wayness property for PUFs similar to the one in the definition of physical one-way functions as proposed by Pappu [104].

Definition 14 A PUF class $\mathcal{P}$ exhibits one-wayness if it is evaluable, and if given a random PUF instance $\mathsf{puf} \in \mathcal{P}$, there exists no efficient algorithm $\mathsf{invert}_{\mathsf{puf}} : \mathcal{Y}_{\mathcal{P}} \to \mathcal{X}_{\mathcal{P}}$ which is allowed to evaluate puf a feasible number of times and for which it holds that

$$\Pr\big(\mathbf{dist}\big[Y \leftarrow \mathsf{puf}(X);\, Y' \leftarrow \mathsf{puf}\big(\mathsf{invert}_{\mathsf{puf}}(Y)\big)\big] > D^{\mathsf{intra}}_{\mathcal{P}}(X)\big) \text{ is low,}$$

for $X \leftarrow \mathcal{X}_{\mathcal{P}}$.

Hence, given a PUF instance and a random response of that instance, there exists no efficient inversion algorithm acting on the PUF instance, that finds a challenge that would produce a response close to the given response. This definition resembles the classic definition of a one-way function in theoretical cryptography, but takes the unreliability and the uniqueness of a PUF instance into account. However, the notion of one-wayness is somewhat ambiguous for PUF constructions since, besides depending on the actual algorithmic complexity of inverting a PUF instance, it also depends very much on the attainable sizes of the challenge and response sets of the PUF constructions. For PUF constructions with a small challenge set, one-wayness is not achievable since the inversion algorithm can easily evaluate every possible challenge and perform an inverse table lookup to invert a given response. On the other hand, if the response set is small, the inversion algorithm can evaluate random challenges on the PUF instance and will quickly encounter one which inverts a given response.

3.2.8 *Tamper Evidence*

Tampering is the act of permanently altering the physical integrity of a system, e.g. of a PUF instance, with the intent of modifying its operation in an unauthorized and possibly harmful manner. It is hence a type of physical transformation which we denote as $\mathsf{puf} \Rightarrow \mathsf{puf}'$ to make clear that physical changes were made. Directed tampering represents a powerful attack against security implementations. It can be used to remove or bypass protection mechanisms and leave the implementation vulnerable, or to obtain information about sensitive internal values and parameters. Protection against tampering is a matter of detecting tampering and providing an appropriate reaction, e.g. clearing confidential data and/or blocking all functionality. In order to detect tampering, a security system needs to have some level of tamper evidence, i.e. a tampering attempt will have an unavoidable and measurable impact on the system. Certain types of PUF constructions, which rely on sensitive measurements of random physical features of an instance, cannot be physically tampered with without significantly changing their challenge-response behavior.

Definition 15 A PUF class $\mathcal{P}$ exhibits tamper evidence if it is evaluable, and if given a random PUF instance $\mathsf{puf} \in \mathcal{P}$, any physical transformation of $\mathsf{puf} \Rightarrow \mathsf{puf}'$ has the effect that

$$\Pr\big(\mathbf{dist}\big[Y \leftarrow \mathsf{puf}(X);\, Y' \leftarrow \mathsf{puf}'(X)\big] > D_{\mathcal{P}}^{\mathsf{intra}}(X)\big) \text{ is high,}$$

and ideally that

$$\Pr\big(\mathbf{dist}\big[Y \leftarrow \mathsf{puf}(X);\, Y' \leftarrow \mathsf{puf}'(X)\big] < D_{\mathcal{P}}^{\mathsf{inter}}(X)\big) \text{ is low.}$$

Informally, tamper evidence means that it is very hard to physically alter a PUF instance without having a noticeable effect on its challenge-response behavior, i.e. an effect which with high probability is larger than the unreliability of the PUF as expressed by the distribution of $D_{\mathcal{P}}^{\mathsf{intra}}$. Ideally, such an alteration even causes the PUF instance to become a completely different one, i.e. the effect on its challenge-response behavior is indiscernible from replacing the PUF instance with a different unique instance as expressed by the distribution of $D_{\mathcal{P}}^{\mathsf{inter}}$.

In a sense, tamper evidence is an orthogonal property to physical unclonability. Physical unclonability states that it is very hard to make two distinct PUF instances more alike than is to be expected from physically different instances. Tamper evidence on the other hand states that it is very hard to physically alter a single PUF instance resulting in a different PUF instance which is more alike to the original than is to be expected from physically different instances. Tamper-evident PUFs are self-protecting in the sense that a tampering attack on their implementation unavoidably and substantially alters their responses, which generally results in a blocked functionality or a loss of secret information. Moreover, they can also be deployed as to make a larger security system tamper-resistant by encapsulating the system in a tamper-evident PUF instance.

3.2.9 PUF Properties Analysis and Discussion

We will now evaluate the properties defined in this section on a representative set of proposed PUF constructions described in Chap. 2, as well as on a number of non-PUF reference cases. This analysis will clarify which of these properties are met by all PUF proposals, which are only met by a few or by none at all, and most importantly, which subset of properties differentiates the PUF and non-PUF constructions.

Representative Subset of Constructions

We have selected a representative set of both non-intrinsic and intrinsic PUF constructions to do this comparative analysis on. We have only selected proposals for which actual implementations have been done and for which experimental data is available. The PUF constructions we will consider are:

1. The optical PUF as proposed by Pappu et al. [105].
2. The coating PUF as proposed by Tuyls et al. [147].
3. The simple arbiter PUF as proposed by Lee et al. [75].
4. The feed-forward arbiter PUF as proposed by Lee et al. [75].
5. The XOR arbiter PUF as proposed by Majzoobi et al. [94].
6. The basic ring oscillator PUF as proposed by Suh and Devadas [136].
7. The enhanced ring oscillator PUF as proposed by Maiti et al. [92].
8. The SRAM PUF as proposed by Guajardo et al. [45].
9. The latch, flip-flop, butterfly and buskeeper PUFs as respectively proposed by Su et al. [135], Maes et al. [84], Kumar et al. [72] and Simons et al. [132]. It is clear that these PUFs will have identical properties since they only differ slightly in their construction details. Therefore we treat them simultaneously.
10. The glitch PUF as proposed by Shimizu et al. [129].
11. The bistable ring PUF as proposed by Chen et al. [22].

To be able to differentiate between PUFs and non-PUF constructions which are used to achieve similar objectives, we have selected a representative set of non-PUF reference constructions. In order to be able to assess the properties of this section on these non-PUF references, we explicitly state what we consider to be the challenge and the response. The non-PUF reference cases we consider are:

1. A true random number generator or TRNG. A TRNG is a process which generates a stream of uniformly random numbers based on measurements of a dynamically random physical process. A TRNG's output is *truly random* since it results from a non-deterministic physical process, as opposed to the output of a mathematical algorithm, known as a seeded *pseudo-random* number generator, which only appears random but can in fact be deterministically calculated if the seed is known. For easy comparison, we say the single challenge of the TRNG is a request for a random number and the response is a fixed-length random output string.
2. A very simple, unsecured radio-frequency identification (RFID) scheme. In this most basic form, an RFID tag is nothing more than a small non-volatile memory which upon deployment is programmed with a unique identifier string. When triggered by a reader, the tag broadcasts this string to identify itself. For easy comparison, we say the single challenge of this RFID scheme is the reader's trigger and the response is the identifier string broadcasted by the tag.
3. An implementation of a secure (unkeyed) cryptographic hash function. We say the challenge is a (fixed-length) message string and the response is the resulting hash digest of that message.

4. An implementation of a secure cryptographic block cipher, with a randomly generated encryption key which is programmed in a non-volatile memory in the implementation. We say the challenge of this block cipher is a single block of message data and the response is the resulting block of ciphertext data.
5. An implementation of a secure cryptographic public-key signature algorithm, with a randomly generated public/private key pair of which the private key is programmed in a non-volatile memory in the implementation. We say the challenge of this signature algorithm is a (fixed-length) message string and the response is the resulting signature on this message.

Comparative Analysis

For every construction listed above, we have assessed whether it meets the different properties discussed in this section. The resulting analysis is shown in Table 3.1. We make four distinctions as to what extent a construction exhibits a certain property:

- **V**: the construction exhibits the property fully or to a large extent
- **X**: the construction does not exhibit the property
- **!**: the construction only exhibits the property under certain conditions
- **?**: it is not clear or unknown whether the construction exhibits the property

Note that the assessments presented in Table 3.1 are only a reflection of the current state of the art, to the best of our knowledge. However, due to the natural progress in mathematical and physical attacks, and in manufacturing techniques, some of these classifications can change over time. We will shortly discuss the presented results for each property and the reasoning behind certain classifications.

Constructibility All considered constructions are constructible since for all of them known implementations exist. However, some of them require more construction effort than others. The optical and coating PUFs rely on explicitly introduced randomness during manufacturing and are therefore non-intrinsic; the other considered PUF constructions are intrinsic. The RFID scheme, the block cipher and the signature algorithm also require explicit programming of a random string.

Evaluability All considered constructions are also evaluable since experimental results are available for all of them. However, some of them require more evaluation effort than others. The optical PUF evaluation procedure is quite elaborate, requiring a laser and a very accurate mechanical positioning system. The SRAM PUF and the flip-flop and buskeeper PUFs require a power cycle to evaluate them, since they rely on the power-up behavior of their construction.

Reproducibility This is the first differentiating property between PUFs and non-PUFs, since it clearly distinguishes the PUF constructions from the TRNG. A TRNG is by definition not reproducible, since this would make it deterministic. The other non-PUF reference cases are perfectly reproducible, i.e. with zero intra-distance. The reproducibility of the PUF proposals varies from more than 25 % average intra-distance for the optical PUF to less than 1 % for the ring-oscillator PUF.

Table 3.1 Properties of a representative subset of PUF and non-PUF constructions

Property	**Optical PUF**	**Coating PUF**	**Simple Arbiter PUF**	**FF Arbiter PUF**	**XOR Arbiter PUF**	**Basic Ring Oscillator PUF**	**Enhanced Ring Oscillator PUF**	**SRAM PUF**	**Flip-flop / Butterfly / Latch / Buskeeper PUF**	**Glitch PUF**	**Bistable Ring PUF**	**TRNG Output**	**RFID Broadcast**	**Cryptographic Hash Function**	**Block Cipher Encryption**	**Public-key Signature**
Constructibility	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V
Evaluability	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V
Reproducibility	V	V	V	V	V	V	V	V	V	V	V	X	V	V	V	V
Uniqueness	V	V	V	V	V	V	V	V	V	V	V	V	V	X	V	V
Identifiability	V	V	V	V	V	V	V	V	V	V	V	X	V	X	V	V
Physical Unclonability	V	V	V	V	V	V	V	V	V	V	V	V	X	X	X	X
Unpredictability	V	V	!	!	?	V	?	V	V	?	?	V	X	X	V	V
Mathematical Unclonability	V	X	X	X	?	X	?	X	X	?	?	V	X	X	!	!
True Unclonability	V	X	X	X	?	X	?	X	X	?	?	V	X	X	X	X
One-wayness	V	X	X	X	X	X	X	X	X	X	X	X	X	V	V	X
Tamper Evidence	V	V	?	?	?	?	?	?	?	?	?	?	!	X	!	!

Legend:

V	*construction exhibits property*	?	*it is unknown/untested whether construction exhibits property (more research required)*
X	*construction does not exhibit property*	!	*construction exhibits property only under certain conditions*

Uniqueness All PUF proposals exhibit uniqueness with average inter-distances of merely 23 % and 38 % respectively for the simple and the feed-forward arbiter PUFs, and very close to 50 % for the other constructions. The TRNG evidently also exhibits ideal uniqueness. Uniqueness is a differentiating feature between PUFs and hash functions. Because the considered hash function has no random elements, every implementation instance will exhibit exactly the same challenge-response behavior. The other three non-PUF constructions are unique if and only if they have been programmed with a unique bit string, which is true with high probability in the regular manufacturing process.

Identifiability This is the combination of reproducibility and uniqueness. TRNGs and hash functions do not meet this property, respectively for not being reproducible and not being unique. All other constructions exhibit identifiability, with for most of them even a large separation between the distributions of intra- and inter-distance, allowing an unambiguous identification based on their responses.

Physical Unclonability As expected, physical unclonability is the great divider between PUF and non-PUF constructions. The uniqueness of all PUF proposals results from random physical processes during their manufacturing process, either implicitly or explicitly, which are so complex that it is considered infeasible to have any meaningful influence on them. Moreover, for all considered PUF constructions, these random processes take effect at microscopic and (deep) submicron levels, which makes it even more technically infeasible to analyse them or exert any control over them. This is in great contrast to the uniqueness of the RFID scheme, the block cipher and the signature algorithm which is based on the programming of a relatively short random bit string. For an adversary which controls the manufacturing process, and hence this programming step, it is not at all difficult to program more than one instance with the same string. For this reason, these constructions are not considered physically unclonable.

Unpredictability Following the modeling attacks on simple and feed-forward arbiter PUFs as discussed in Sect. 2.4.1, these two constructions only remain unpredictable as long as an adversary does not learn enough challenge-response pairs to accurately train its model. The actual number of unpredictable responses depends on the details of the implementation and on the state of the art in modeling attacks, but this can be very limited: modeling attacks have been presented which achieve better than random accuracy after less than 100 training responses and improve to near-perfect accuracy when more responses are learned. For the XOR arbiter PUF with a sufficient number of XOR-ed arbiters (≥ 5), no effective modeling attacks are yet known, and the same holds for the enhanced ring oscillator PUF, the glitch PUF and the bistable ring PUF, although for none of these four PUFs can strong claims of independence between response bits be made. For the optical PUF, reasonable arguments are presented by Škorić et al. [133], Tuyls et al. [146] that modeling attacks are computationally infeasible. For the remaining PUFs, response bits are produced by physically distinct elements, which is a strong motivation to assume that different responses are independent and hence inherently unpredictable. Consequently, no modeling attacks are known for these PUFs. All the non-PUF constructions except for the RFID scheme and the hash function exhibit unpredictability, the TRNG because of random physical influences and the block cipher and the signature scheme because of a secret element and computational complexity arguments. The RFID scheme is trivially predictable since it only has one single fixed response. The hash function construction contains no unique element (key) and is easily predicted.

Mathematical Unclonability Simple and feed-forward arbiter PUF constructions are not mathematically unclonable since they are only conditionally unpredictable. The coating PUF, the ring oscillator PUF, the SRAM PUF and the four other bistable memory element PUFs do not exhibit mathematical unclonability because the size of their challenge sets is small, which means they can be fully evaluated to produce a lookup table as a trivial mathematical clone. For the XOR arbiter PUF, the enhanced ring oscillator PUF, the glitch PUF and the bistable ring PUF, no practical mathematical cloning attacks are known, but more research is required before any strong claims can be made. Škorić et al. [133], Tuyls et al. [146] argue that the optical PUF is to a large extent mathematically unclonable, since even if (hypothetically) a mathematical model of an optical PUF instance could be created, it would be computationally too complex to be evaluated. However, this turns out to be very implementation-dependent, as Rührmair [113] states that a variant of an optical PUF with reduced randomness can in fact be modeled. Finally, the block cipher and the signature algorithm are assumed to be mathematically unclonable under the condition that an adversary is not able to extract the secret key during its unlimited physical access to an instance. This implies that, besides being algorithmically secure, the implementations of these cryptographic primitives also need to be physically protected, e.g. against side-channel and fault attacks, and against reverse-engineering.

True Unclonability This is the combination of physical and mathematical unclonability. The optical PUF is the only one of the considered PUF constructions for which strong claims of true unclonability can be made.

One-Wayness Besides the optical PUF, none of the other studied PUF proposals can be considered one-way since they either have a small challenge or a small response set. Pappu [104] presents strong arguments why his optical PUF construction does exhibit one-wayness, hence being labelled a physical one-way function. If the block cipher is algorithmically secure and its key remains secret, it is considered to be hard to invert. The same holds for the unkeyed hash function. A public-key signature algorithm is not guaranteed to be one-way, since everyone with knowledge of the public key can verify the signature, which possibly involves recovering the signed message.

Tamper Evidence A certain level of tamper evidence was only experimentally demonstrated for optical PUFs by Pappu [104] and for coating PUFs by Tuyls et al. [147]. For all other proposed PUF constructions no results on tamper evidence, neither in the positive nor in the negative sense, are known. Hence no sensible claims

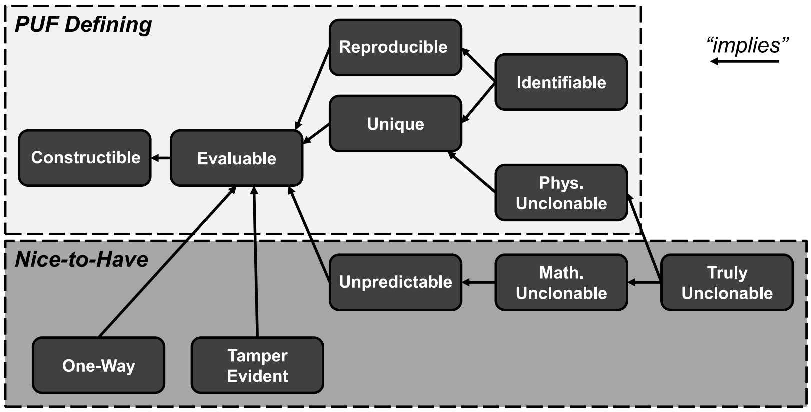


Fig. 3.1 Relations between the different described PUF properties and an indication of the PUF defining properties

on tamper evidence can be made for them. For TRNGs, tamper evidence is not well defined. The remaining non-PUF proposals are not inherently tamper-evident, but can be implemented in a tamper-evident fashion by applying tamper detection measures.

3.2.10 Discussion on PUF Properties

PUF Defining Properties

Looking at the results of the comparative property study in Table 3.1, and assuming these are representative for all PUF and non-PUF constructions, we can determine the properties which are *defining* for PUFs, i.e. which properties do all PUFs meet that distinguish them from all conceivable non-PUF constructions? The answer to this question turns out to be *identifiability*, and to a larger extent *physical unclonability*, which is in fact the enforcement of uniqueness in the presence of an adversary with control over the instance creation process. Note that these two properties imply that PUFs are also constructible, evaluable, reproducible and unique. The relations between all discussed properties, as well as an indication of the PUF defining properties, are shown in Fig. 3.1. Based on this analysis, we tentatively propose a definition of a PUF class.

Definition 16 A class of physical entities with a challenge-response functionality is called a PUF class if it exhibits identifiability (cf. Definition 9) and physical unclonability (cf. Definition 10).

In the light of this definition, we argue that the acronym 'PUF' as standing for *physically unclonable* function, i.e. with the qualifier 'physically' reflecting on 'unclonable', not on 'function', is particularly fitting for the concept. This strong emphasis on physical unclonability being the core PUF property also implicates that if at one point the physical unclonability of a particular PUF construction is disputed, e.g. due to the natural advancement of manufacturing techniques, it will cease to be a PUF.

Nice-to-Have PUF Properties

Since only identifiability and physical unclonability actually define PUFs, the remaining properties of unpredictability, mathematical and true unclonability, one-wayness and tamper evidence are merely desirable extras, but are not guaranteed for any PUF construction. In fact, there currently seems to be only one PUF proposal which meets all these properties, and that is the optical PUF as proposed by Pappu [104]. This observation, combined with the fact that it was one of the very first PUF proposals, grants the optical PUF the status of *prototype PUF*. All following PUF constructions aim to achieve as many of the desirable properties of the optical PUF as possible, but at the same time aim to provide more integrated implementations.

Since these remaining properties are only nice-to-have qualities, they cannot simply be assumed to be present in any PUF. This means that PUF designers need to present strong arguments, preferably of an experimental nature, if they claim any of these extra properties for their PUF proposal. (In fact, they also need to show identifiability and physical unclonability to demonstrate that their proposal is a PUF to begin with.) For developers of PUF-based applications, this entails that they need to state explicitly if, and to what extent, they rely on any of these nice-to-have properties, since it means that not all PUF constructions can be used for their application.

Improving PUF Properties

From Table 3.1 it is clear that mathematical unclonability, and in consequence true unclonability, and one-wayness are hard to come by for most PUF proposals. When these properties are required in a PUF-based application, they can be provided by extending the raw physical PUF instance with an algorithmic primitive exhibiting these properties, and enforcing that the resulting PUF-system only be evaluated with this primitive. This is what was defined as a *controlled PUF* in Sect. 2.5.2.

3.3 Formalizing PUFs

In Sect. 3.2, we described meaningful properties of PUFs, and after a broad comparison identified the ones which define a PUF. Based on this analysis, we proceed towards a strictly formal description of PUFs and their key properties. First, we study earlier proposed attempts at formally describing PUFs and point out how they fall short. Next we discuss our approach in setting up the framework and in Sect. 3.3.3 we introduce the basic primitives of the framework itself. Using the introduced framework, we propose formal definitions respectively for the notions of robustness, physical unclonability and unpredictability. The rationale behind the definitions of all concepts and properties in this section is to provide a meaningful formal model for both hardware engineers (developing PUFs) and cryptographers (deploying PUFs).

Background The development of the formal framework for physical functions and the formal definitions of their properties as initially proposed in [3] and discussed in this section are the shared results of numerous fruitful discussions and an intense research collaboration between the author and Prof. Frederik Armknecht (Universität Mannheim), Prof. François-Xavier Standaert (Université catholique de Louvain), Christian Wachsmann and Prof. Ahmad-Reza Sadeghi (both Technische Universität Darmstadt).

3.3.1 Earlier Formalization Attempts

Throughout PUF literature, many authors have attempted to generalize the concept of a PUF in a more or less formal definition, mainly as a means to highlight the advantageous properties of a simultaneously proposed new PUF construction. We briefly introduce these definitions and point out why we believe none of them captures the full spectrum of proposed PUFs and their properties, either by being too restrictive, i.e. excluding certain PUFs, or by being too ad hoc, i.e. listing perceived and even assumed properties of certain PUFs instead of providing a generalizing model. A similar overview and discussion has been presented by Rührmair et al. [115]. However, we do not completely follow all their arguments and moreover point out why the new models they propose are still insufficient.

Another approach toward defining the functionality of a PUF comes from the theoretical corner. Theorists, in an attempt to deploy PUFs in their algorithms and protocols, provide rather rigid formal descriptions of PUFs on which they can built security reductions. We also discuss these proposals and their drawbacks.

Physical One-Way Functions

To the best of our knowledge, the first generalizing definition of the PUF concept is given by Pappu [104], based on the properties of his optical PUF construction. He focuses on the one-wayness property of this construction and labels it a *physical one-way function* (POWF). The first part of the definition of a POWF states that it is a deterministic physical interaction that is evaluable in constant time but cannot be inverted by a probabilistic polynomial time adversary with a non-negligible probability. The second part of the definition focusses on the unclonability of the POWF: both simulating a response and physically cloning the POWF should be hard. The POWF definition is solely based on the optical PUF, which at that time was the only known PUF. As other PUFs were introduced shortly after, it became clear that this definition was too stringent, in particular regarding the one-wayness assumption. While the optical PUF has very large challenge and response sets, many of the later introduced PUFs do not. For these constructions, one-wayness does not hold any longer since inverting such a PUF with non-negligible advantage becomes trivial, as discussed in Sect. 3.2.7. It is also noteworthy that, as pointed out by Rührmair et al. [115], for most PUF-based security applications, one-wayness is not a required condition. A final issue with the POWF definition is that it lacks any notion of noise; in fact it even describes a POWF as a deterministic interaction. This is contradicted by the fact that virtually all PUF proposals, including the optical PUF, produce noisy responses due to uncontrollable physical influences affecting a response evaluation.

Physical Random Functions

With the introduction of delay-based intrinsic PUFs, Gassend et al. [42], propose the definition of *physical random functions* to describe PUFs. In brief, a physical random function is defined as a function embodied by a physical device which is *easy to evaluate* but *hard to predict* from a polynomial number of challenge-response observations. Note that this definition replaces the very stringent one-wayness condition from POWFs with a more relaxed unpredictability condition. However, the presentation of modeling attacks on simple arbiter PUFs by Lee et al. [75] and on more elaborate arbiter PUFs by Rührmair et al. [119] demonstrate a significantly reduced unpredictability of these types of PUFs. Moreover, the later introduced memory-based intrinsic PUFs only possess at most a polynomial number of challenges and hence do not classify as physical random functions since they can be easily modeled through exhaustive readout. Finally, the definition of physical random functions as proposed by Gassend et al. [42] also does not capture the possibility of noisy responses.

Weak and Strong PUFs

With the introduction of memory-based intrinsic PUFs, Guajardo et al. [45] further refine the formal specification of a PUF. They describe PUFs as inherently unclonable physical systems with a challenge-response behavior. It is *assumed* that: (i) responses to different challenges are independent of each other, (ii) it is difficult to come up with responses which have not been observed before, and (iii) tampering with a PUF instance substantially changes its challenge-response behavior. For the first time, it is made explicit that PUF responses are observed as noisy measurements. This definition also comes with a division of *strong* and *weak* PUFs, depending on how many challenge-response pairs an adversary is allowed to obtain in order to model the PUF. If the number is exponentially large in some security parameter, the PUF is called a strong PUF; otherwise the PUF is called weak. It can be argued that some of the assumptions made in this description do not have a solid experimental basis, in particular regarding tamper evidence, which has not been tested in practice for any of the intrinsic PUF proposals. Also, strong PUFs are difficult to characterize in general, as the idea of a security parameter is specific to each PUF instance, and no practical procedure is proposed to exhibit the required exponential behavior in practice.

Rührmair et al. [115] build upon the distinction between strong and weak PUFs from Guajardo et al. [45] and redefine both notions in terms of a security game with an adversary. Weak PUFs are called *obfuscating PUFs* and are basically considered as physically obfuscated keys, as described in Sect. 2.5.1. The main statement in the definition of obfuscating PUFs is that an adversary cannot learn the key after having had access to the PUF for a limited amount of time. Strong PUFs are defined similarly, but here the adversary needs to come up with the response to a randomly chosen challenge after having had access to the PUF and a PUF oracle for some limited time. Some issues are again left unresolved in this formalization: first, despite building upon the work of Guajardo et al. [45], responses are not considered to be noisy. Next, the use of a PUF oracle in the definition of a strong PUF seems questionable. It is argued that this oracle is introduced to circumvent practical access restrictions to the PUF. However, if a PUF-based system is secured against any attacks possible with the current state of technology, the access to such an oracle is an unrealistic advantage to the adversary, which weakens the proposed definition.

PUFs and PUF-PRFs

In [2] we have introduced a first formal PUF model which starts from a theoretical application perspective, rather than from a practical construction-based perspective as most earlier proposals. In [2], we aim to use a PUF as part of a cryptographic algorithm, i.c. a block cipher, and for that goal the previously discussed definitions prove to be insufficient. We explicitly make a distinction between algorithmic and physical properties of a PUF. From the algorithmic side, a PUF is said to be a noisy function for which the distribution of responses is indistinguishable from a random distribution with a certain amount of min-entropy. From the physical side, a PUF is assumed to be physically unclonable and tamper-evident. A PUF-PRF is then defined as a PUF-based system with pseudo-random function-like qualities. We already pointed out the lack of experimental proof for tamper evidence of intrinsic PUFs in practice, and the same argument applies to this definition. The description of uniqueness and unpredictability by means of min-entropy provides convenient qualities for the theoretical application of PUFs, but as it turns out it is hard to give strong proof of min-entropy levels for actual PUF constructions. Contrarily to most of the previous definitions, PUFs are explicitly defined as noisy functions with a strictly bounded noise magnitude.

PUFs in the Universal Composition Framework

Brzuska et al. [15] continue the theoretical application viewpoint approach toward defining PUFs, and propose a very formal definition which allows them to model PUF functionality in the universal composition framework as proposed by Canetti [20]. As in our definition from [2], they define PUFs based on response distributions having a certain level of min-entropy and also incorporate a noise threshold. However, the presented formulation is utterly theoretical, which makes it unattractive for practice-oriented designers of PUF constructions. Consequentially, there is a significant probability that an actual PUF construction which lives up to this stringent definition will never be proposed. Moreover, the strong min-entropy assumptions on the PUF responses rule out many known PUF constructions and put strong restrictions on others, leading to practical inefficiency.

3.3.2 Setup of the Formal Framework

Objective

The basic objective of the formal framework presented in [3], and which is discussed in detail in the following, is to provide a workable model for both practice-oriented PUF designers as well as theory-oriented cryptographers deploying PUFs in algorithms and protocols. Ideally, the model is sufficiently realistic, capturing measurable properties of actual PUF proposals, while at the same time providing sufficient formalism and rigor to allow theoretical security reductions of systems deploying a PUF as a primitive. Such a framework could serve as an *interface* between hardware engineers and theoretical cryptographers, which would be very beneficial for the continued successful deployment of PUFs in security systems.

Approach

The approach we take is to start from a minimalistic axiomatic framework to describe *physical functions* and increment it in a flexible manner; first, by hierarchically expanding the notion of a physical function to more extensive constructions, and secondly by defining modular properties of these constructions within the framework.

The particular difficulty experienced by formal modeling attempts of PUFs is dealing with the physical aspect. It is hard to argue about the security properties of a physical object because they typically cannot be captured by classical cryptographic notions such as security parameters or computationally hard problems. To isolate this difficulty, we capture the physical aspect at the lowest formal level when we axiomatically describe the notion of a physical function. All higher-level properties can then be defined using the formalism introduced to describe physical functions. The particular properties which we consider are robustness, physical unclonability and unpredictability. Note that what we call a physical function is a very general concept, which is broader than PUFs, and physical unclonability is merely one possible property of a physical function.

3.3.3 Definition and Expansion of a Physical Function

Physical Function (pf)

A physical function pf consists of a physical component p and an evaluation procedure $\mathsf{Eval}_{a_{\mathsf{ev}}}$. A physical component can be physically stimulated, resulting in a measurable effect. The evaluation procedure $\mathsf{Eval}_{a_{\mathsf{ev}}}$ translates the physical stimulus and resulting measurement into digital forms, respectively called the challenge x and the response y of the physical function. The exact challenge-response behavior of a pf is determined by both the static and the dynamical physical states of its physical component, and by an evaluation parameter a_{ev} which controls the challenge and response translation, e.g. the quantization step size of an analog measurement.

Definition 17 A physical function pf is a probabilistic procedure

$$\mathsf{pf}_{\mathsf{p},a_{\mathsf{ev}}} : \mathcal{X} \to \mathcal{Y},$$

consisting of an evaluation procedure Eval acting upon a physical component p:

$$y \leftarrow \mathsf{pf}_{\mathsf{p},a_{\mathsf{ev}}}(x) = \mathsf{Eval}_{a_{\mathsf{ev}}}(\mathsf{p}; x).$$

When the physical component and the evaluation parameter are clear from the context, we simply write pf instead of $\mathsf{pf}_{\mathsf{p},a_{\mathsf{ev}}}$.

Extraction Algorithm (Extract)

A physical function is not a function in the classical sense since, when challenged with the same challenge x twice, it may produce different responses. This is an effect of the response representing a measurement of a physical component whose physical state is partially dynamic, e.g. as a result of non-deterministic random physical noise during the measurement. However, for many applications this is an undesirable feature of a physical function, and it is dealt with by an appropriate extraction algorithm $\mathsf{Extract}_{a_{\mathsf{ex}}}$ for which it is possible to guarantee the reproducibility of its output. Many instantiations of extraction algorithms exist, including the seminal *fuzzy extractor* as proposed by Dodis et al. [32, 33]. As with physical functions, we describe extraction algorithms as generically as possible to allow the greatest possible flexibility of the framework. An extraction procedure extracts an output $z \in \mathcal{Z}$ from a response y of a pf, and in the process it can also consume and/or produce some additional side information which we call *helper data* and denote as $w \in \mathcal{W}$. We also introduce an extraction parameter a_{ex} which is used to exactly specify all the deployment details of the extractor.

Definition 18 An extraction algorithm Extract is a probabilistic procedure

$$\mathsf{Extract}_{a_{\mathsf{ex}}} : \mathcal{Y} \times \mathcal{W} \to \mathcal{Z} \times \mathcal{W},$$

which operates in one of two modes depending on the format of the presented helper data:

$$\begin{aligned} \textbf{[setup]} \quad & (z, w) \leftarrow \mathsf{Extract}_{a_{\mathsf{ex}}}(y, \epsilon), \\ \textbf{[reconstruction]} \quad & \bigl(z', w' = w\bigr) \leftarrow \mathsf{Extract}_{a_{\mathsf{ex}}}\bigl(y', w \neq \epsilon\bigr), \end{aligned}$$

with ϵ denoting the empty string.

When the extraction parameters are clear from the context, we simply write $\mathsf{Extract}$ instead of $\mathsf{Extract}_{a_{\mathsf{ex}}}$.

In setup mode, when no helper data is presented as an input ($w = \epsilon$), the extraction algorithm produces an output z and helper data w. In reconstruction mode, another possibly noisy evaluation of the response y' is presented together with helper data w which was produced in an earlier setup mode of the extractor. The extraction algorithm re-extracts the output z' and additionally outputs the unchanged helper data w. The power of most extraction algorithms is that, under certain conditions on the pf response distribution, they succeed in recreating exactly the same output in both setup and reconstruction mode: $z = z'$, given that the helper data generated by the setup mode is used during reconstruction mode. Besides this reconstruction property, an extraction algorithm can also provide guarantees about the randomness of its output z. The actual implementation of the extraction algorithm, as is the case for the physical function, is left up to the practical developer. The generic nature of the definition allows a wide variety of extractor implementations, including using no extractor at all by making it the identity function.

Physical Function System (pfs)

In many application scenarios, the use of an extraction algorithm is indispensable, and by consequence a user will only be aware of the challenge provided to the physical function and the output generated by the extractor. The existence of an intermediate physical function response is transparent to him. Additionally, the relevant security notions in such a scenario will be determined by the combination of both the used physical function and the deployed extraction algorithm. For these reasons, it makes sense to abstract away the separate notions of a physical function and an extraction algorithm and consider their combination as a single building block. We call such a combination a *physical function system* pfs.

Definition 19 A physical function system pfs is a probabilistic procedure

$$\mathsf{pfs}_{\mathsf{p},a_{\mathsf{ev}},a_{\mathsf{ex}}} : \mathcal{X} \times \mathcal{W} \to \mathcal{Z} \times \mathcal{W},$$

consisting of the concatenation of a physical function (cf. Definition 17) with an extraction algorithm (cf. Definition 18):

$$\bigl(z, w'\bigr) \leftarrow \mathsf{pfs}_{\mathsf{p},a_{\mathsf{ev}},a_{\mathsf{ex}}}(x, w) = \mathsf{Extract}_{a_{\mathsf{ex}}}\bigl(\mathsf{pf}_{\mathsf{p},a_{\mathsf{ev}}}(x), w\bigr).$$

When the physical component and the parameters are clear from the context, we write $\mathsf{pfs}(x, w)$ instead of $\mathsf{pfs}_{\mathsf{p},a_{\mathsf{ev}},a_{\mathsf{ex}}}(x, w)$.

We believe that, from a theoretical perspective, it is easier to reason about physical function systems than to deal with the technical peculiarities of physical functions and extractors. Therefore we will define formal properties over physical function systems, rather than over physical functions, and we only refer to the underlying physical functions and extractors when necessary.

Physical Function Infrastructure ($\mathcal{F}$)

The physical component p in a physical function pf is the result of a physical creation process $\mathsf{Create}_{a_{\mathsf{cr}}}$. The exact manifestation of a created physical component is determined by stochastic influences during the creation process, and by a controlled deterministic creation parameter a_{cr} which defines the full details of the creation process.

Definition 20 For a fixed tuple of parameters $(a_{\mathsf{cr}}, a_{\mathsf{ev}}, a_{\mathsf{ex}})$, the physical function infrastructure $\mathcal{F}_{(a_{\mathsf{cr}},a_{\mathsf{ev}},a_{\mathsf{ex}})}$ refers to the creation process $\mathsf{Create}_{a_{\mathsf{cr}}}$ and the set of all physical function systems consisting of the extraction algorithm $\mathsf{Extract}_{a_{\mathsf{ex}}}$, and a physical function $\mathsf{pf}_{\mathsf{p},a_{\mathsf{ev}}}$ with a physical component created by $\mathsf{Create}_{a_{\mathsf{cr}}}$:

$$\mathcal{F}_{(a_{\mathsf{cr}},a_{\mathsf{ev}},a_{\mathsf{ex}})} \stackrel{\triangle}{=} \left(\mathsf{Create}_{a_{\mathsf{cr}}}, \{\mathsf{pfs}_{\mathsf{p},a_{\mathsf{ev}},a_{\mathsf{ex}}} : \mathsf{p} \leftarrow \mathsf{Create}_{a_{\mathsf{cr}}}\}\right).$$

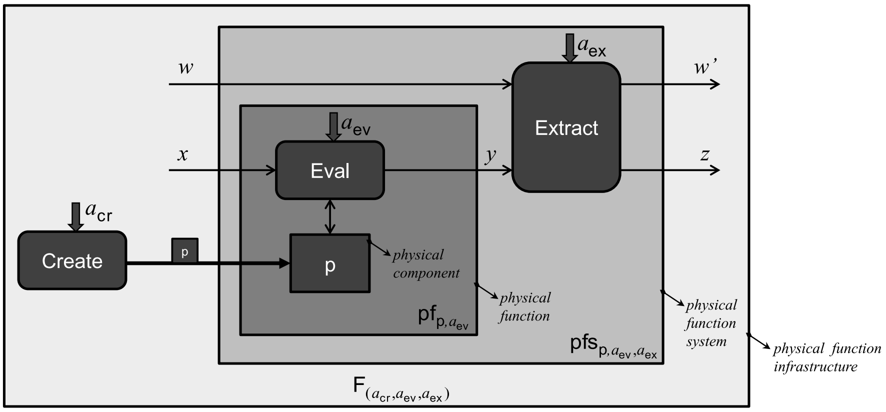
Finally, a family of physical function infrastructures is defined as a generalization of a physical function infrastructure for more than one single creation parameter a_{cr}:

$$\mathcal{F}_{(\mathcal{A}_{\mathsf{cr}},a_{\mathsf{ev}},a_{\mathsf{ex}})} \stackrel{\triangle}{=} \{\mathcal{F}_{a_{\mathsf{cr}},a_{\mathsf{ev}},a_{\mathsf{ex}}} : a_{\mathsf{cr}} \in \mathcal{A}_{\mathsf{cr}}\}.$$

If the parameters a_{ev} and a_{ex} are clear from the context, we simply write $\mathcal{F}_{a_{\mathsf{cr}}}$ and $\mathcal{F}_{\mathcal{A}_{\mathsf{cr}}}$. We use a family of physical function infrastructures to express the control one has over the creation procedure $\mathsf{Create}_{a_{\mathsf{cr}}}$, by being able to pick the creation parameter a_{cr} from a certain set $\mathcal{A}_{\mathsf{cr}}$.

Overview

In Fig. 3.2, we schematically show the concept of a physical function as the combination of a physical component and an evaluation procedure. Also shown is the expansion to a physical function system, by combining it with an extraction procedure, and further to a physical function infrastructure, by combining it with a creation procedure. Each of the three proposed procedures is possibly probabilistic and is for the remainder fully determined by its inputs and the respective parameters a_{ev}, a_{ex} and a_{cr}.



a_{ex}
w
w'
Extract
a_{ev}
x
Eval
y
z
a_{cr}
Create
p
p
physical component
$\mathsf{pf}_{\mathsf{p},a_{\text{ev}}}$
physical function
physical function system
$\mathsf{pfs}_{\mathsf{p},a_{\text{ev}},a_{\text{ex}}}$
physical function infrastructure
$\mathsf{F}_{(a_{\text{cr}},a_{\text{ev}},a_{\text{ex}})}$

Fig. 3.2 Schematic overview of the concept of a physical function and its extension to a physical function system and a physical function infrastructure

3.3.4 Robustness of a Physical Function System

Robustness as defined in this section is the formal counterpart of reproducibility as described in Sect. 3.2.2, with the difference that reproducibility allows a certain error between evaluations as long as it is small, whereas robustness describes error-free reconstructions. In practice, an appropriate extraction algorithm is able to transform a reproducible PUF into a robust physical function system.

The robustness of a physical function system is defined as the probability that an output z produced during setup mode can later be reproduced exactly in reconstruction mode.

Definition 21 The challenge robustness of pfs with respect to $x \in \mathcal{X}$ is defined as

$$\rho_{\mathsf{pfs}}(x) \triangleq \Pr\big((z, w) \leftarrow \mathsf{pfs}(x, w) : (z, w) \leftarrow \mathsf{pfs}(x, \epsilon)\big).$$

When considering a subset of challenges $\mathcal{X}' \subseteq \mathcal{X}$, the following related robustness notions can be defined:

- Minimum robustness of pfs with respect to $\mathcal{X}'$:

$$\rho_{\mathsf{pfs}}^{\min}(\mathcal{X}') \triangleq \min_{x \in \mathcal{X}'} \{\rho_{\mathsf{pfs}}(x)\}.$$

  This is useful when one requires every considered challenge to exhibit a minimal level of robustness.
- Average robustness of pfs with respect to $\mathcal{X}'$:

$$\rho_{\mathsf{pfs}}^{\text{av}}(\mathcal{X}') \triangleq \sum_{x \in \mathcal{X}'} \rho_{\mathsf{pfs}}(x) \cdot \Pr(x \leftarrow \mathcal{X}').$$

From a practical viewpoint it is often sufficient to have a high enough average robustness.

This notion of robustness can even be extended further to physical function infrastructures. The average robustness of a physical function infrastructure $\mathcal{F}$ is defined as:

$$\rho^{\mathrm{av}}_{\mathcal{F}}(\mathcal{X}') \triangleq \sum_{\mathsf{pfs}\in\mathcal{F}} \rho^{\mathrm{av}}_{\mathcal{F}}(\mathcal{X}') \cdot \Pr(\mathcal{F} \leftarrow \mathsf{Create}).$$

When $\mathcal{X}' = \mathcal{X}$ or when the content of $\mathcal{X}'$ is clear from the context, we simply write $\rho^{\mathrm{min}}_{\mathsf{pfs}}$, $\rho^{\mathrm{av}}_{\mathsf{pfs}}$ and $\rho^{\mathrm{av}}_{\mathcal{F}}$.

3.3.5 Physical Unclonability of a Physical Function System

Defining a (Physical) Clone

Before we define unclonability, we first need to agree on what we consider to be a clone. Intuitively, we consider two instances clones if they show ‘similar behavior’. However, there are a number of technical details which have to be taken into account in order to turn this intuition into a formal definition.

First of all, we need to make explicit that we consider *physical* unclonability, and therefore also only *physical* clones. Given the definition of a physical function system, there are a number of non-physical ways one can think of to create ‘similar instances’. For example, a physical function system deploying an extraction algorithm which outputs a fixed constant is trivial to clone, but it is obvious that we do not consider this a physical clone. A similar argument holds for the evaluation procedure of the physical function, e.g. imagine an evaluation procedure which quantizes a physical measurement into a zero bit response, which is basically a fixed value. We explicitly only consider physical clones by stating that two physical function systems which are considered clones can only differ in their physical component p, while their evaluation and extraction procedures and parameters need to be identical.

Secondly, there are many levels of ‘similarity’: two physical function systems can be ‘less different than expected’ or they can be truly ‘indistinguishable’. This needs to be captured by a quantitative parameter. Also, we need to consider similarity with respect to a subset of challenges. If two physical function systems happen to coincide on a subset of challenges which is critical for a particular application, they need to be considered clones, even if they show completely different behavior on the remainder of challenges.

Finally, when attempting to formalize the notion of ‘similarity’ for physical function systems, we will run into robustness again. How does one define similarity with respect to a physical function system which even by itself does not always generate similar outputs? To tackle this issue, we are guided by two intuitive arguments: (i) every physical function system pfs should be a clone of itself (except for not deploying a different physical component), and (ii) a clone of a physical function system pfs cannot be more similar to pfs than pfs is to itself, as expressed by its robustness. In other words, the robustness of pfs is a natural upper bound for how similar a clone can be to pfs. Therefore, we express the similarity of a clone to pfs relative to its robustness.

Definition 22 For a fixed tuple of parameters $(a_{\mathsf{ev}}, a_{\mathsf{ex}})$, let $\mathsf{pfs}(=\mathsf{pfs}_{\mathsf{p},a_{\mathsf{ev}},a_{\mathsf{ex}}})$ and $\mathsf{pfs}'(=\mathsf{pfs}_{\mathsf{p}',a_{\mathsf{ev}},a_{\mathsf{ex}}})$ be two physical function systems which are identical except for their physical components, $\mathsf{p} \neq \mathsf{p}'$. We say pfs is a δ-clone of pfs' with respect to $\mathcal{X}' \subseteq \mathcal{X}$, if $\forall x \in \mathcal{X}'$ it holds that

$$\Pr\big((z, w) \leftarrow \mathsf{pfs}'(x, w) : (z, w) \leftarrow \mathsf{pfs}(x, \epsilon)\big) \geq \delta \cdot \rho_{\mathsf{pfs}}(x),$$

with $0 \leq \delta \leq 1$. In shorthand notation, we write: $\mathsf{pfs} \overset{\delta;\mathcal{X}'}{\equiv} \mathsf{pfs}'$.

By $\mathsf{p} \neq \mathsf{p}'$ we mean that p and p' are distinct physical entities, i.e. occupying different positions in space-time, but they are allowed to be physically similar to any level of precision. Note that, except for their not deploying different physical components, every physical function system is a $(\delta = 1, \mathcal{X}' = \mathcal{X})$-clone of itself.

Defining Physical Unclonability

Now that we have a formal definition of a clone, we can define physical unclonability by formalizing the statement: ‘it is difficult to produce a clone’. However, again some technicalities need to be considered before a formal description can be presented.

We first need to specify the capabilities of the adversary A that is trying to produce a clone. In practice, such an adversary will have access to a number of executions of the creation procedure of physical components. Possibly, it even has an amount of control over it, i.e. it can influence the physical processes taking place during creation within certain boundaries. We capture this formally by allowing the adversary to select the creation parameter a_{cr} from a particular subset $\mathcal{A}_{\mathsf{cr}}$. We use the notion of a family of physical function infrastructures $\mathcal{F}_{\mathcal{A}_{\mathsf{cr}}}$ to describe this. The adversarial model is described by means of a security game between the adversary and a creation oracle.

We also need to distinguish between two variants of unclonability:

- *Existential unclonability* means that it is hard to create a pair of physical function systems such that one is a clone of the other.
- *Selective unclonability* means that given a particular physical function system, it is hard to create a second one which is a clone of the first one.

Note that in general, existential unclonability implies selective unclonability and is therefore a stronger security notion. We will give a formal definition for existential unclonability, but the same approach as presented here can be applied to describe selective unclonability.

To formally define (existential) unclonability (or cloning resistance), we first describe the adversary model A by means of a *cloning game* $\mathbf{Game}^{\mathsf{clone}}_{\mathsf{A}}(\mathcal{A}_{\mathsf{cr}}, q)$:

- In the cloning game $\mathbf{Game}^{\mathsf{clone}}_{\mathsf{A}}(\mathcal{A}_{\mathsf{cr}}, q)$, an adversary is allowed to make up to q queries to a creation oracle $\mathsf{O}^{\mathsf{Create}}_{\mathcal{A}_{\mathsf{cr}}}$, with $q \geq 2$.
- The creation oracle $\mathsf{O}^{\mathsf{Create}}_{\mathcal{A}_{\mathsf{cr}}}$ expects a creation parameter a_{cr} as query input. If $a_{\mathsf{cr}} \in \mathcal{A}_{\mathsf{cr}}$, then the oracle invokes the physical component creation procedure with the queried parameter to create a single physical component $\mathsf{Create}_{a_{\mathsf{cr}}} \to \mathsf{p}$ and answers the query with p.
- The adversary is allowed to adaptively change the creation parameter a_{cr} of its queries.
- When the game ends, the adversary is required to output a pair of physical components $(\mathsf{p}, \mathsf{p}')$ both of which it received as a query reply from the creation oracle during the game.

Definition 23 A family of physical function instantiations $\mathcal{F}_{(\mathcal{A}_{\mathsf{cr}}, a_{\mathsf{ev}}, a_{\mathsf{ex}})}$ is (γ, δ, q)-cloning-resistant with respect to $\mathcal{X}' \subseteq \mathcal{X}$ if for every probabilistic polynomial time adversary A it holds that:

$$\Pr\big(\mathsf{pfs}_{\mathsf{p}, a_{\mathsf{ev}}, a_{\mathsf{ex}}} \overset{\delta;\mathcal{X}'}{\equiv} \mathsf{pfs}'_{\mathsf{p}', a_{\mathsf{ev}}, a_{\mathsf{ex}}} : (\mathsf{p}, \mathsf{p}') \leftarrow \mathbf{Game}^{\mathsf{clone}}_{\mathsf{A}}(\mathcal{A}_{\mathsf{cr}}, q)\big) \leq \gamma.$$

The level of control an adversary has over the creation process will to a large extent determine the cloning resistance of a physical function infrastructure family. As explained, the influence an adversary has over the creation is represented by $\mathcal{A}_{\mathsf{cr}}$. We distinguish a special case when $\mathcal{A}_{\mathsf{cr}} = \{a_{\mathsf{cr}}\}$, i.e. the creation process is fixed. This is typically so for the genuine manufacturer of the physical function systems and therefore we call this the *honest manufacturer* adversary model. Note that even the honest manufacturer can coincidentally create clones but this should only happen with low probability. In all other cases with more than one element in $\mathcal{A}_{\mathsf{cr}}$, it means that the manufacturer is deliberately influencing the creation process in order to create a clone. This is called the *malicious manufacturer* adversary model.

3.3.6 Unpredictability of a Physical Function System

When using a physical function system pfs in a security application, the unpredictability of its output is the most basic expected security requirement. In classic cryptography, unpredictability is a well-established concept, e.g. for random functions, and it expresses the difficulty of predicting an unobserved output of a function after having observed different function evaluations. Due to the peculiarities of physical function systems, as discussed in detail in the previous paragraphs, the classical definition of unpredictability does not directly apply. We will adapt it in an appropriate manner as to make it capture the notion of unpredictability for physical function systems.

Types of Unpredictability

We distinguish between two different types of unpredictability for physical function systems: unpredictability with respect to different outputs on the same system, and unpredictability with respect to the same outputs on different systems. The first type is typically important when the physical function system is used as a challenge-response entity, e.g. in an authentication protocol. It would be highly undesirable if the response in the next run of the protocol could be predicted based on previously observed runs of the protocol. The second type is mainly of significance when the physical function system is used as a secure storage mechanism, e.g. to generate cryptographic keys. In that case, the independence of outputs from different physical function systems is of the utmost importance to ensure the randomness of the derived keys. Note that the first type of unpredictability is a direct extension of the classical unpredictability notion to physical function systems, and a theoretical definition will be a formal variant of the unpredictability property of PUFs as discussed in Sect. 3.2.5. The second type of unpredictability, on the other hand, could be regarded as a generalization of the uniqueness property of PUFs as discussed in Sect. 3.2.3. Whereas uniqueness only requires that different PUF instances produce sufficiently different responses, this type of unpredictability additionally requires a more stringent apparent independence between the outputs of different physical function systems. The formalization we propose next captures both types of unpredictability in a single definition, as well as the continuum of intermediate cases.

Defining Unpredictability for Physical Function Systems

We again use a game-based approach to describe a model for the adversary A. We distinguish between a weak and a strong prediction game, based on the control the adversary has over picking the physical function systems and challenges which are evaluated.

- The *weak prediction game* $\mathbf{Game}_{\mathsf{A}}^{\mathsf{predict;weak}}(\mathcal{P}_{\mathsf{learn}}, \mathcal{P}_{\mathsf{chal}}, q)$ is a game between an adversary A and an evaluation oracle $\mathsf{O}^{\mathsf{Eval}}_{\mathcal{P}_{\mathsf{learn}},\mathcal{P}_{\mathsf{chal}}}$ which takes place in two phases: a learning phase and a challenge phase.
- During the learning phase, A is allowed to query the oracle up to q times. When queried, $\mathsf{O}^{\mathsf{Eval}}_{\mathcal{P}_{\mathsf{learn}},\mathcal{P}_{\mathsf{chal}}}$ randomly selects $\mathsf{pfs}_i \overset{\$}{\leftarrow} \mathcal{P}_{\mathsf{learn}}$ and $x_i \overset{\$}{\leftarrow} \mathcal{X}$ and evaluates $(z_i, w_i) \leftarrow \mathsf{pfs}_i(x_i, \epsilon)$. For each of these q queries, the adversary A is allowed to observe the tuple $(\mathsf{pfs}_i, x_i, z_i, w_i)$.
- During the challenge phase, $\mathsf{O}^{\mathsf{Eval}}_{\mathcal{P}_{\mathsf{learn}},\mathcal{P}_{\mathsf{chal}}}$ randomly selects $\mathsf{pfs} \overset{\$}{\leftarrow} \mathcal{P}_{\mathsf{chal}}$ and $x \overset{\$}{\leftarrow} \mathcal{X}$, making sure that the combination (pfs, x) was never evaluated during

the learning phase, and evaluates $(z, w) \leftarrow \mathsf{pfs}(x, \epsilon)$. The adversary A is now allowed to observe the tuple (pfs, x, w).
- At the end of the game, the oracle outputs the tuple (pfs, x, z) and the adversary outputs a prediction z'.

The *strong prediction game* $\mathbf{Game}_{\mathsf{A}}^{\mathsf{predict;strong}}(\mathcal{P}_{\mathsf{learn}}, \mathcal{P}_{\mathsf{chal}}, q)$ is completely equivalent to the weak prediction game, only now physical function systems and challenges are no longer randomly selected by the oracle, but are adaptively queried by the adversary. In the learning phase, the adversary queries the oracle with up to q tuples $(\mathsf{pfs}_i \in \mathcal{P}_{\mathsf{learn}}, x_i, w_i)$ and learns the full evaluations $(z_i, w'_i) \leftarrow \mathsf{pfs}_i(x_i, w_i)$ from the oracle. In the challenge phase, the adversary queries the oracle with a single tuple $(\mathsf{pfs} \in \mathcal{P}_{\mathsf{chal}}, x, w)$ such that the combination (pfs, x) was never queried during the learning phase. The oracle evaluates $(z, w') \leftarrow \mathsf{pfs}(x, w)$ but A can only observe w' this time.

Definition 24 Let $\mathcal{P}_{\mathsf{learn}}$ and $\mathcal{P}_{\mathsf{chal}}$ be subsets containing physical function systems from the same physical function infrastructure $\mathcal{F}$. We say the physical function systems in $\mathcal{P}_{\mathsf{chal}}$ are (λ, q)-weakly unpredictable with respect to $\mathcal{P}_{\mathsf{learn}}$ if for every probabilistic polynomial time adversary A, it holds that:

$$\Pr\big(z = z' : \big((\mathsf{pfs}, x, z), z'\big) \leftarrow \mathbf{Game}_{\mathsf{A}}^{\mathsf{predict;weak}}(\mathcal{P}_{\mathsf{learn}}, \mathcal{P}_{\mathsf{chal}}, q)\big) \leq \lambda \cdot \rho_{\mathsf{pfs}}(x).$$

Similarly, we say the physical function systems in $\mathcal{P}_{\mathsf{chal}}$ are (λ, q)-strongly unpredictable with respect to $\mathcal{P}_{\mathsf{learn}}$ if for every probabilistic polynomial time adversary A, it holds that:

$$\Pr\big(z = z' : \big((\mathsf{pfs}, x, z), z'\big) \leftarrow \mathbf{Game}_{\mathsf{A}}^{\mathsf{predict;strong}}(\mathcal{P}_{\mathsf{learn}}, \mathcal{P}_{\mathsf{chal}}, q)\big) \leq \lambda \cdot \rho_{\mathsf{pfs}}(x).$$

Note that $\mathcal{P}_{\mathsf{chal}}$ and $\mathcal{P}_{\mathsf{learn}}$ do not need to be mutually exclusive and can even be identical. In fact, if $\mathcal{P}_{\mathsf{chal}} = \mathcal{P}_{\mathsf{learn}} = \{\mathsf{pfs}\}$, i.e. we only consider the unpredictability of a single pfs with respect to itself, then Definition 24 closely resembles the classical notion of unpredictability of a random function.

3.3.7 Discussion

Most of the concepts and properties defined in this section are more rigid formalizations of concepts and properties which we introduced earlier in a more intuitive way, respectively in Sects. 2.2.1 and 3.2.

- The formally defined notion of a physical function infrastructure ($\mathcal{F}$) coincides almost completely with what we, rather intuitively, have called a PUF class ($\mathcal{P}$) in Sect. 2.2.1. Both describe a set of instantiations and a creation procedure (Create).
- A PUF instance (puf), described in Sect. 2.2.1 as having a *physical state* which can be measured by an evaluation procedure (Eval), is equivalent to a formal physical function (pf) consisting of a *physical component* (p) and the same evaluation procedure.

- Concrete extractor constructions have not yet been described, but are treated in detail in Chap. 6. It is evident that they are captured formally by the rather generic extraction procedure introduced in this section. The formal concept of a physical function system coincides with the concatenation of a PUF with an extractor implementation.
- The formally defined property of robustness of a physical function system is an extension of the earlier introduced PUF property of reproducibility (cf. Definition 7), taking into account the effect of the extractor.
- Equivalently, the formal definition of cloning resistance is the extension of physical unclonability (cf. Definition 10), related to the formal version of robustness.
- The formal notion of unpredictability is defined in a broad sense, considering both unpredictability with respect to the same instance as well as regarding other instances. In that aspect, the formal unpredictability is to be considered as the formalization of the earlier introduced unpredictability notion (cf. Definition 11, describing unpredictability with regard to different responses on the same PUF instance) in combination with a generalized version of uniqueness (cf. Definition 8, describing *differentness* of responses with regard to other PUF instances).

The resemblance between these formal definitions and the rather intuitive concepts and properties which we have defined earlier makes clear that the proposed framework is of a significant practical value. This is strengthened by the fact that most of the intuitive properties have been experimentally verified for many PUF implementations and can hence be directly translated to their formal counterparts. Evidently, to do this, the effect of the extractor needs to be taken into account.

3.4 Conclusion

Following an extensive study of 11 meaningful PUF properties on a representative subset of PUF constructions, and on a reference set of non-PUF constructions, we are able to extract the defining properties of a PUF: *identifiability and physical unclonability*, and by implication constructibility, evaluability, reproducibility and uniqueness. The remaining properties of unpredictability, mathematical and true unclonability, one-wayness and tamper evidence are classified as *nice-to-have*, but are not strictly required for a construction to be called a PUF. This also means that many PUF proposals in fact do not exhibit these nice-to-have properties, or at the very best it remains as yet unclear whether they do. In this light, the optical PUF as proposed by Pappu et al. [105] serves as an exemplary prototype PUF, meeting all mentioned properties to a major extent. Most intrinsic PUFs fall short on the property of mathematical unclonability, and the existence of a *strong intrinsic PUF* with motivated and verified security guarantees remains an open question. One-wayness turns out to be a particularly unfitting property for intrinsic PUFs, which is partially caused by the ambiguity of considering the one-wayness of PUFs. Tamper evidence, while often proclaimed as one of the major advantages of using PUFs, remains a question mark for all intrinsic PUF proposals, as no experimental results on any construction are known.

Based on these results and with the aim of formalizing these particularly interesting properties, we introduce a framework for working with PUFs (and physical functions in general) in a theoretical security setting. Using the introduced primitives in this framework, we formally define robustness, physical unclonability and unpredictability of a physical function.