Towards a Unified Security Model for Physically Unclonable Functions*

Frederik Armknecht[1], Daisuke Moriyama[2], and Ahmad-Reza Sadeghi[3] and Moti Yung[4]

[1] University of Mannheim, Germany
[2] NICT, Japan
[3] TU Darmstadt, Germany
[4] Google and Columbia University, USA

Abstract. The use of Physically Unclonable Functions (PUFs) in cryptographic protocols attracted an increased interest over recent years. Since sound security analysis requires a concise specification of the alleged properties of the PUF, there have been numerous trials to provide formal security models for PUFs. However, all these approaches have been tailored to specific types of applications or specific PUF instantiations. For the sake of applicability, composability, and comparability, however, there is a strong need for a unified security model for PUFs (to satisfy, for example, a need to answer whether a future protocol requirements match a new and coming PUF realization properties).
In this work, we propose a PUF model which generalizes various existing PUF models and includes security properties that have not been modeled so far. We prove the relation between some of the properties, and also discuss the relation of our model to existing ones.

Keywords: Physically unclonable function, security model, specifications

1 Introduction

Physically Unclonable Functions (PUFs) are functions represented by physical objects which are mainly provided by unavoidable arbitrary variations during the manufacturing process. PUFs can be used to secure secret key generation and key management as an alternative to achieving this by dedicated (more expensive) security processors, such as Trusted Platform Module (TPM) and employing random number generation. Currently, there are three major research topics regarding PUFs:

1. Hardware: Proposing a new construction of PUF or evaluating existing PUFs based on implementation (FPGA, ASIC, etc.) [13, 14, 19, 20, 23, 27].

* The preliminary version of this paper is presented in CT-RSA 2016 [3] and this full version includes security proofs for implecation and separation results among the security notions for PUFs.

2. Protocol Design: Considering a PUF as an abstract building block and proposing new cryptographic primitives or protocols [5, 7, 8, 15, 22, 24, 30, 32].
3. Modeling: Investigating theoretical perspectives on PUFs and describing a security model [1, 2, 7, 13, 14, 26, 27].

In particular, we note that there have been, obviously, multiple attempts to come up with security models for PUFs, these often aimed for specific types of applications and/or PUF hardware types. This results in the very unsatisfying situation that, up to now, there is no one-for-all PUF model (or super-model) which covers all desired properties. This deficiency has some serious consequences. For example, protocols where security has been shown under different models cannot be easily combined without requiring a new security analysis. In the worst case, formalizations may be even incompatible, which would demand a complete reevaluation of these parts. Another problem is that protocol designers face the challenge of choosing the "right" security model among the existing ones, and mapping models to devices is not often clear. In fact, as we will discuss, there exist PUF-based protocols which require a selection of security properties that are not covered by a single model yet.

Contribution

In this work, we aim at closing this gap, by comparing the various existing models, describing a new security model which unifies and extends them, and confront its properties against hardware devices and protocols in the field.

First, the new model covers the most relevant security properties of PUFs. The overall situation is depicted in Table 1. Here, the columns display the considered security properties. In the upper part of the table, we mark for a variety of security models, and which of these properties is covered by the respective model. Each model, indeed, covers some (or all) of the following notions: sufficient min-entropy of the outputs, one-wayness, unforgeability, and unclonability. To motivate the necessity of a unifying model, in the lower part we give examples of three previously published protocols, which security properties need to hold for these protocols to be secure. As one can observe, some models would not be suitable to analyze the security of some of the protocols. In fact only the model of Brzuska et al. [7] includes all four properties.

Unfortunately, even this model is not sufficiently comprehensive. For example, the RFID authentication scheme described in [30] requires that the PUF outputs are pseudorandom, a property not included into any of these models. Thus, our model does not only unify these models but formalizes three novel security properties: indistinguishability, pseudorandomness, and tamper resilience.

A further extension given by our model is the output distributions of PUFs. Due to the fact that PUF outputs are noisy, all models include the notion of *intra-distance* of outputs. This refers to the distance (with respect to an according metric) between several outputs of the same PUF on the same input. However, we argue that in addition two types of *inter-distance* should be part of a comprehensive security model. By *inter-distance I* we consider the variation of

Table 1. Comparison of several existing PUF security model and overview of required security properties for different protocol examples. One can analyze other PUF-based schemes or protocols so that which properties are required with this table.

Security Model	Min-entropy	One-wayness	Unforge-ability	Unclon-ability	Indistin-guishability	Pseudo-randomness	Tamper-resilience
Pappu [27]	-	X	X	-	-	-	-
Gassend et al. [13]	-	-	X	-	-	-	-
Guajardo et al. [14]	-	-	X	-	-	-	-
Armknecht et al. [2]	X	-	-	X	-	-	-
Armknecht et al. [1]	X	-	X	X	-	-	-
Brzuska et al. [7]	X	X	X	X	-	-	-
Maes [21]	-	X	X	X	-	-	-
Ours	X	X	X	X	X	X	X
Example Protocols	**Min-entropy**	**One-wayness**	**Unforge-ability**	**Unclon-ability**	**Indistin-guishability**	**Pseudo-randomness**	**Tamper-resilience**
Challenge-Response [27]	-	-	X	-	-	-	-
PUF-PRF [2]	X	-	X	X	-	-	-
RFID Authentication [30]	-	-	-	X	-	X	-

outputs of a *single* PUFs when queried on *multiple* input while *inter-distance II* is about the distance between the outputs of *multiple* PUFs on the *same* input. So far, inter-distance I has been covered only by the model of Gassend et al. [13], while inter-distance II is only part of the model by Maes [21].

Since all these properties are covered by our model, it represents the most comprehensive model so far. We discuss the relation of our model to existing works in more detail at the end of this paper.

Important note: We have to stress that we do not claim that a single PUF should meet all these properties. This is clearly not the case. However, each of the properties covered in our model has been considered in previous work or is natural to be considered (e.g., tamper resilience). In that sense, we see our model as the most general (i.e., a super-model) and flexible one, that when given a PUF-based protocol requirements, allows to express the necessary security properties.

Outline Section 2 summarizes the preliminaries. Section 3 describes our model in detail, and explains relations between the covered security properties. Section 4 compares our model to related work, while Section 5 concludes the paper.

2 Notations

For a probabilistic machine or algorithm A, the term $A(x)$ denotes the random variable of the output of A on input x. $a \leftarrow A(x)$ indicates the event that A outputs a on input x the value a. When A is a set, $y \xleftarrow{\cup} A$ means that y is uniformly selected from A. $|A| \leq \mathsf{poly}(x)$ indicates that the number of elements in A is polynomially bounded by x. When the parameter x is clear from the

context, we omit it. When a is a value, $y := a$ denotes that y is set as a. For two values a, a', the expression $\mathsf{Dist}(a, a')$ denotes the distance between a and a' according to some metrics (e.g., Hamming distance, edit distance). $\tilde{H}_\infty(A)$ indicates the min-entropy of A and $\tilde{H}_\infty(A \mid B)$ evaluates the conditional min-entropy of A given B.

3 Security Model: Properties and Their Relationships

In this section, we describe our model. We start with a specification of the overall system and then formalize various properties of PUFs which are security relevant. A PUF is a probabilistic mapping $f : \mathcal{D} \to \mathcal{R}$ where $\mathcal{D}$ is a domain space and $\mathcal{R}$ is an output range of PUF f. The creation of a PUF is formally expressed by invoking a manufacturing process $\mathcal{MP}$. That is a $\mathcal{MP}$ is a randomized procedure which takes inputs from a range of parameters and outputs a new PUF. We do not specify the input range of $\mathcal{MP}$ in purpose as it strongly depends on the concrete PUF but also on the considered attacker model. For example, in the weakest attacker model the input range of $\mathcal{MP}$ is empty. This would model the case that there is one legitimate process for creating the PUF and an attacker can only invoke exactly this procedure. The other extreme is that $\mathcal{MP}$ is a kind of "universal creation process" where for any product an according parameter input does exist. In general, one may imagine that $\mathcal{MP}$ represents a class of creating processes which is parameterized. Next, we formalize the security properties with respect to a given security parameter λ and PPT attackers (polynomial in λ).

3.1 Output Distribution

Due to the fact that PUFs have noisy outputs, considering the output distribution is important. In the following, we specify four different requirements with respect to different aspects of the output distribution. Depending on the concrete application, one would have to choose a PUF where some or all of these conditions are met. We first give a formal definition and explain its rationale afterwards. The following definitions are parametrized by some thresholds δ_i, the number of iterations t, the number of inputs ℓ, the number of devices n, a negligible function $\epsilon(\cdot)$, and the security parameter λ.

Intra-Distance Requirement: Whenever a single PUF is repeatedly evaluated with a fixed input, the maximum distance between the corresponding outputs is at most δ_1. That is for any created PUF $f \leftarrow \mathcal{MP}(param)$ and any $y \in \mathcal{D}$, it holds that

$$\Pr\left[\max(\{\mathsf{Dist}(z_i, z_j)\}_{i \neq j}) \leq \delta_1 \mid y \in \mathcal{D}, \{z_i \leftarrow f(y)\}_{1 \leq i \leq t}\right] = 1 - \epsilon(\lambda).$$

Inter-Distance I Requirement: Whenever a single PUF is evaluated on different inputs, the minimum distance among them is at least δ_2. That is for a created PUF $f \leftarrow \mathcal{MP}(param)$ and for any $y_1, \ldots, y_\ell \in \mathcal{D}$, we have

$$\Pr\left[\min(\{\mathsf{Dist}(z_i, z_j)\}_{i \neq j}) \geq \delta_2 \,\middle|\, \begin{matrix} y_1, \ldots, y_\ell \in \mathcal{D}, \\ \{z_i \leftarrow f(y_i)\}_{1 \leq i \leq \ell} \end{matrix}\right] = 1 - \epsilon(\lambda).$$

Inter-Distance II Requirement: Whenever multiple PUFs are evaluated on a single, fixed input, the minimum distance among them is at least δ_3. That is for any created PUF $f_i \leftarrow \mathcal{MP}(param)$ for $1 \leq i \leq n$ and any $y \in \mathcal{D}$, we have

$$\Pr\left[\min(\{\mathsf{Dist}(z_i, z_j)\}_{i \neq j}) \geq \delta_3 \,|y \in \mathcal{D}, \{z_i \leftarrow f_i(y)\}_{1 \leq i \leq n}\right] = 1 - \epsilon(\lambda).$$

Min-Entropy Requirement: Whenever multiple PUFs are evaluated on multiple inputs, the min-entropy of the outputs is at least δ_4, even if the other outputs are observed. Let $z_{i,j} \leftarrow f_i(y_j)$ be the output of a PUF f_i on input y_j where $f_i \leftarrow \mathcal{MP}(param)$. Then

$$\Pr\left[\tilde{H}_\infty(z_{i,j} \mid \mathcal{Z}_{i,j}) \geq \delta_4 \,\middle|\, \begin{array}{c} y_1, \ldots, y_\ell \in \mathcal{D}, \\ \mathcal{Z} := \{z_{i,j} \leftarrow f_i(y_j)\}_{1 \leq i \leq n, 1 \leq j \leq \ell}, \\ \mathcal{Z}_{i,j} := \mathcal{Z} \setminus \{z_{i,j}\} \end{array}\right] = 1 - \epsilon(\lambda)$$

holds for sufficiently large δ_4.

Definition 1. *A PUF $f : \mathcal{D} \to \mathcal{R}$ has* $(\mathcal{MP}, t, n, \ell, \delta_1, \delta_2, \delta_3, \epsilon)$-variance *if the PUF's output has inter and intra distances as described above, parameterized by* $(\mathcal{MP}, t, n, \ell, \delta_1, \delta_2, \delta_3)$.

Definition 2. *A PUF $f : \mathcal{D} \to \mathcal{R}$ has* $(\mathcal{MP}, n, \ell, \delta_4, \epsilon)$-min-entropy *if the PUF satisfies the min-entropy requirement explained above.*

The *intra-distance* and the two metrics of *inter-distance* are very important notions, crucial to ensure the correctness of schemes built on top of the PUF. For example, if $\delta_1 \geq \delta_2$ then outputs from the same inputs may exhibit a higher distance than outputs coming from different inputs. Similarly, $\delta_1 \geq \delta_3$ would result in the situation that outputs of the same PUF have a larger distance than outputs of other PUFs. This is, for example, critical when PUFs are used as authenticating devices. Therefore, $\delta_1 < \delta_2$ and $\delta_1 < \delta_3$ are necessary conditions to allow for a clear distinction between different inputs and different PUFs. These are fundamental issues to assure the *uniqueness* for each output.

One popular method to assert the uncertainty of the PUF's output is the notion of min-entropy. For example the min-entropy is an important aspects if combined with a *fuzzy extractor* [12] to ensure outputs with a sufficient level of randomness can be reconstructed nonetheless. Consequently, Bzruska et al. [7] included the notion of min-entropy in their model, but limited their definition to the case that the inputs have all a certain Hamming distance, which we omit in our model. Since the restriction of the inputs requires extra cost for the scheme layer itself, and correlated inputs may influence the min-entropy evaluation, we define the min-entropy for arbitrary chosen $y_i \in \mathcal{D}$. Furthermore, our min-entropy evaluation is more general than [7], so that outputs from other devices are also included to evaluate the conditional entropy. This is useful when we consider a multi-party setting where each party holds his own PUF.

Next, we provide formal security definitions for PUF properties that are based on security notions from "classical" cryptographic primitives. Throughout the

rest of the paper, we assume that the number of PUFs created by a specific parameter via Create is polynomially bounded in λ and we simply denote the upper bound as n. Similarly, the $\mathsf{Response}$ query issued by a malicious adversary to obtain the PUF's response is also polynomially bounded. We also assume that intra-distance δ_1 is strictly smaller than any of the inter-distances (δ_2, δ_3) (except with negligible probability ϵ).

3.2 One-wayness

One of the most basic security requirements in cryptography is one-wayness. This is formalized by the following game between a challenger and an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

Setup. The challenger selects a manufacturing process $\mathcal{MP}$ and initial parameter $param$. The challenger sends $(1^\lambda, \mathcal{MP}, param)$ to adversary $\mathcal{A}_1$. In addition, the challenger creates a list List which is initially empty and initializes two counters (c_0, c_1).

Phase 1. $\mathcal{A}_1$ can adaptively issue the following oracle queries.

- When $\mathcal{A}_1$ issues $\mathsf{Create}(param')$, the challenger checks $param'$. If $param' = param$, the challenger increments c_0 and creates a new PUF $f_{c_0} \leftarrow \mathcal{MP}(param)$. If $param' \neq param$ and $param'$ is a valid input to the manufacturing process, the challenger increments c_1 and invokes $f'_{c_1} \leftarrow \mathcal{MP}(param')$. Otherwise, the challenger responds with $\perp$.
- When $\mathcal{A}_1$ sends $\mathsf{Response}(b, i, y_j)$ with $b \in \{0, 1\}$, the challenger proceeds as follows. If $b = 0$ (indicating that a correctly constructed PUF shall be queried) and if $i \leq c_0$, the challenger responds $z_{i,j} \leftarrow f_i(y_j)$. If $b = 1$ and $i \leq c_1$, the challenger responds $z'_{i,j} \leftarrow f'_i(y_j)$. Otherwise, the challenger outputs $\perp$.

Challenge. When $\mathcal{A}_1$ finishes Phase 1, $\mathcal{A}_1$ sends an index $i^* \leq c_0$ to the challenger and outputs state information st. Then the challenger selects $y^* \xleftarrow{\mathsf{U}} \mathcal{D}$ and responds $z^* \leftarrow f_{i^*}(y^*)$ to $\mathcal{A}$.

Phase 2. Given z^* and st, $\mathcal{A}_2$ continuously issues the oracle query as Phase 1.

Guess. Finally, $\mathcal{A}_2$ outputs y_1^*.

The advantage of the adversary for the above game is defined by

$$\mathsf{Adv}^{\mathsf{OW}}_{\mathcal{A}}(\lambda, \delta_1) := \Pr[y^* = y_1^*] - (\ell + 1)/|\mathcal{D}|$$

where ℓ denotes the number of queries the adversary issued to the i^*-th PUF. The adversary wins the above game if $\mathsf{Adv}^{\mathsf{OW}}_{\mathcal{A}}(\lambda, \delta_1) > 0$ holds with non-negligible probability in λ.

In Phase 1 and 2, the adversary can submit $\mathsf{Create}(param')$ to create a new PUF. If $param' = param$, a PUF is created by the default parameter originally chosen by the challenger. Otherwise, a PUF is created with a different parameter specified by the adversary to generate a malicious PUF [9,26] or bad PUF [11, 28]. A malicious PUF may leak extra information to the adversary.

This is necessary as in general, one cannot exclude that an attacker could learn valuable information from evaluating PUF which are created with (possibly only slightly) different parameters. The adversary can obtain the output of PUFs via oracle query regardless of the parameter setting whenever the PUF has been created. We note that the attack target, chosen in the challenge phase and evaluated in the guess phase, is a PUF created by the default parameter $param$. As we will see later, the concept of malicious PUFs, i.e., PUFs being created by different parameters, are also useful to discuss the relationship to the notion of unclonability. $(\ell+1)/|\mathcal{D}|$ gives the probability that the adversary trivially breaks the one-wayness with random guess when we faithfully cover the noise from the PUF; more detailed discussion is appeared in Appendix A.

Definition 3. *A PUF provides $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-one-wayness if for any PPT adversary $\mathcal{A}$, $\Pr[\mathsf{Adv}^{\mathsf{OW}}_{\mathcal{A}}(\lambda, \delta_1) > 0] \leq \epsilon(\lambda)$ holds.*

3.3 Unforgeability

Many PUF-based protocols base their security on the assumption that estimating the output of a PUF should not be possible without having access to the device. While several previous works call this notion as *unpredictability*, we refer to this property as *unforgeability*. The main reason is that, as we will show, it shares many similarities with the typical security notions in the context of digital signature schemes or MACs, being Universal Unforgeability (UUF) and Existential Unforgeability (EUF). Both notions are considered in the context of different attack types: Key Only Attack (KOA), Known Message Attack (KMA), and Chosen Message Attack (CMA). In some cases, One Time (OT) security is also considered which refers to the case that the involved oracle can be queried only once.

In our model, we adopt these established security notions for PUFs. The EUF-CMA security game against a PUF is described by the following:

Setup. The challenger proceeds as the setup phase in the one-wayness game and sends $(1^\lambda, \mathcal{MP}, param)$ to adversary $\mathcal{A}$.
Learning. $\mathcal{A}$ can adaptively issue oracle queries (Create and $\mathsf{Response}$) as defined in the one-wayness game.
Guess. After the learning phase, $\mathcal{A}$ outputs (i^*, y^*, z^*).

We disallow the adversary to submit $\mathsf{Response}(param, i^*, y^*)$ in the learning phase. The advantage of the adversary is defined by

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A}}(\lambda, \delta_1) := \Pr[\mathsf{Dist}(z^*, f_{i^*}(y^*)) \leq \delta_1] - |\mathcal{Z}'|/|\mathcal{R}|$$

where f^*_i has been produced by a challenger in the learning phase and $\mathcal{Z}' := \{z \mid z_{i^*} \leftarrow f_{i^*}(y^*), \mathsf{Dist}(z_{i^*}, z) \leq \delta_1\}$. We say that the adversary wins the unforgeability game iff $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A}}(\lambda, \delta_1) > 0$ holds with non-negligible probability in λ.

A similar definition can be found in [1] but their security model considers only PUFs which are combined with a fuzzy extractor. We do not make any

assumption on a post-processing mechanism and consider the security issue for the PUFs itself. Therefore, we do not evaluate the equality but (appropriate) distance between z^* and $f_{i^*}(y^*)$ (interestingly, the existing security models except [1] only consider the equality against the stand-alone PUFs). Since we adopt the intra-distance notion here, there are $|\mathcal{Z}'|$ candidates for $f_{i^*}(y^*)$ in $\mathcal{R}$. Hence, the advantage is defined as the probability to output a candidate minus the probability to simply pick a random element of this set.

Definition 4. *A PUF provides $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-EUF-CMA security if for any PPT adversary $\mathcal{A}$, $\Pr[\mathsf{Adv}_{\mathcal{A}}^{\mathsf{EUF\text{-}CMA}}(\lambda, \delta_1) > 0] \leq \epsilon(\lambda)$ holds.*

3.4 Unclonability

As the name physically unclonable function indicates, an important assumption with respect to a PUF is that it should be hard for an adversary to come up with two PUFs that exhibit quite similar input-output behavior. We capture this by an unclonability game, formalized as follows:

Setup. The challenger proceeds as the setup phase in the one-wayness game and sends $(1^\lambda, \mathcal{MP}, param)$ to adversary $\mathcal{A}$.

Learning. $\mathcal{A}$ can adaptively issue the oracle queries (Create and $\mathsf{Response}$) as defined in the one-wayness game.

Guess. After the learning phase, $\mathcal{A}$ outputs a triple of the form (i^*, b, j^*) with $b \in \{0, 1\}$ and $(b, i^*) \neq (b', j^*)$.

The goal of the attacker is to create a clone to a PUF which stems from the set of PUFs that have been created under the parameters $param$. We refer to these as the *original* parameters. The first entry i^* of the output refers to the i^*-th PUF within this set. The other two parameters (b, j^*) are interpreted as follows. If $b = 0$ then it refers to j^*-th PUF created under the original parameters $param$ otherwise to the j^*-th PUF created under the modified parameters. Let f_{i^*} and f'_{j^*} denote these two PUFs. The adversary wins the unclonability game if PUF f'_{j^*} performs sufficiently similar to f_{i^*}. More formally, the advantage of the adversary is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Clone}}(\lambda, \delta_1) := \Pr[\forall y \in \mathcal{D}, \mathsf{Dist}(f_{i^*}(y), f'_{j^*}(y)) \leq \delta_1].$$

Definition 5. *A PUF provides $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-*unclonability *if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Clone}}(\lambda, \delta_1) \leq \epsilon(\lambda)$ holds.*

Recall that an adversary may use parameters $param'$ for the manufacturing process that are different to the originally used parameters $param$. However, she is only successful if she can clone a PUF that results from the original manufacturing process. On the other hand, the clone itself may result from different parameters $params'$. This has some fundamental consequences. For example, when $|\mathcal{D}| \in \mathsf{poly}(\lambda)$ holds, the adversary can learn all input-output pairs $\{(y_j, z_j)\}_j$ in the learning phase and select $param'$ such that the input-output behavior includes the complete lookup table provided by $\{(y_j, z_j)\}_j$. This

means $(\mathcal{MP}, n, |\mathcal{D}|, \delta_1, \epsilon)$-unclonability cannot be satisfied in such cases. Various memory-based PUFs belong to this class. We stress, however, that this does not mean that such PUFs are of no value, but rather that such PUFs need to be protected by additional measures.

The above definition aims to comprehensively capture the notion of unclonability. To this end, we have to consider two relaxed notions of unclonability. One approach is that the adversary may only create PUFs according to the original manufacturing process, i.e., $param' = param$ in all queries. We call this variant as *target unclonable*. Another way to cover a relaxed notion of unclonability is that we explicitly restrict the upper bound of oracle queries the adversary issues in the learning phase as $\ell < |\mathcal{D}|$. Since $|\mathcal{D}| \in \mathsf{poly}(\lambda)$ holds for memory-based PUFs, it is useful to consider this restriction. We call this variant as *restricted unclonable*.

Observe that our model covers scenarios like building attacks [29, 31] and fault injection attacks [25]. The supervised learning in the machine learning attack analyzes a set of training data as input and estimates an unobserved output, so it is considered as an attack for the EUF-KMA security.

3.5 Indistinguishability

For many cryptographic schemes and protocols, the notion of indistinguishability is fundamental to providing security or privacy. Although it is useful for designers to capture the notion that a PUF's output is indistinguishable from another output, former models ignored this aspect and mainly concentrate on the unforgeability. Consider a simple challenge-response authentication performed by a PUF's input-output pair. The unforgeability against the PUF provides the security against impersonation attack, but the privacy aspect cannot be argued with this notion only. When a PUF satisfies indistinguishability, it means, in principle, that no one can deduce from observed output which PUF has been in use. Therefore, the notion of indistinguishability for PUFs is important with respect to *privacy-preserving* protocols. The indistinguishability game between a challenger and adversary $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ is defined as follows:

Setup. The challenger proceeds as the setup phase in the one-wayness game and sends $(1^\lambda, \mathcal{MP}, param)$ to adversary $\mathcal{A}$.

Phase 1. $\mathcal{A}_1$ can adaptively issue the oracle queries (Create and $\mathsf{Response}$) as defined in the one-wayness game.

Challenge. The adversary submits two tuples (i_0^*, y_0^*) and (i_1^*, y_1^*) which are not issued as $\mathsf{Response}(param, i_0^*, y_0^*)$,$\mathsf{Response}(param, i_1^*, y_1^*)$ in Phase 1. Then the challenger flips a coin $b \xleftarrow{\mathsf{U}} \{0,1\}$ and responds $z_b^* \leftarrow f_{i_b^*}(y_b^*)$ to the adversary.

Phase 2. $\mathcal{A}_2$ receives st and can continuously issue ($\mathsf{Create}, \mathsf{Response}$) except $\mathsf{Response}(param, i_0^*, y_0^*)$ and $\mathsf{Response}(param, i_1^*, y_1^*)$.

Guess. Finally, the adversary outputs a guess b'.

The adversary wins the indistinguishability game if $b' = b$ holds with probability more than $1/2$.

While the PUF is not a deterministic function, the adversary can estimate the challenger's coin if he can obtain $f_{i_0^*}(y_0^*)$ or $f_{i_1^*}(y_1^*)$ by checking the distance from z_b^*. Thus we cannot allow the adversary to issue $\mathsf{Response}(param, i_0^*, y_0^*)$ nor $\mathsf{Response}(param, i_1^*, y_1^*)$. Instead, $\mathcal{A}$ can choose $i_0^* = i_1^*$ to distinguish the output difference from one device or $y_0^* = y_1^*$ to consider the output variance between two devices with same input. The advantage of the adversary in the above indistinguishability-based game is defined by

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IND}}(\lambda) := |2 \cdot \Pr[b' = b] - 1|.$$

Definition 6. *A PUF satisfies* $(\mathcal{MP}, n, \ell, \epsilon)$-indistinguishablility *if for any PPT adversary* $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IND}}(\lambda) \leq \epsilon(\lambda)$ *holds.*

3.6 Pseudorandomness

Some protocols consider a PUF as a kind of physical pseudorandom function that cannot be shared simultaneously by two different parties (e.g., [30]). In fact, depending on how sensitive the PUF behavior is with respect to the physical state, such assumptions may be justified. In any case, a comprehensive model should cover a notion of pseudorandomness. Our definition is based on the pseudorandomness game described below:

Setup. The challenger proceeds as the setup phase in the one-wayness game and sends $(1^\lambda, \mathcal{MP}, param)$ to adversary $\mathcal{A}$. In addition, the challenger flips a coin $b \xleftarrow{\mathsf{U}} \{0,1\}$, creates a list List which is initially empty and prepares counter (c_0, c_1) and truly random function RF, i.e., a random oracle.

Learning. The adversary can issue (Create and $\mathsf{Response}$) queries as defined in the one-wayness game. When the challenger receives a $\mathsf{Response}(param', i, y_j)$ query, the challenger performs the following :

- If $param' \neq param$ or $b = 1$, performs as in the one-wayness game. When $param' = param$ and $i \leq c_0$, responds with $z_{i,j} := f_i(y_j)$. When $param' \neq param$ and $i \leq c_1$, respond $z_{i,j} := f'_i(y_j)$. In other cases, respond $\perp$.
- If $param' = param$ and $b = 0$, the challenger inputs (i, y_j) to RF and obtains $z'_{i,j} \in \mathcal{R}$. Then he selects some random noise and applies it to $z'_{i,j}$ to derive $z_{i,j}$ which satisfies $\mathsf{Dist}(z_{i,j}, z'_{i,j}) \leq \delta_1$. If $i \leq c_0$, respond $z_{i,j}$. Otherwise, output $\perp$.

Guess. Finally, $\mathcal{A}$ outputs a guess b'.

The adversary wins the pseudorandomness game iff $b' = b$.

The main difference from the canonical pseudorandom function is that the challenger does not directly hands outputs $z'_{i,j}$ which came from the truly random function but adds some noise bounded by δ_1. This additional procedure is critical to emulate the actual PUF's behavior from intra-distance perspective. Our description is more suitable to minimize the gap between the real output and ideal output. Even if $b = 0$, the challenger selects the same value $z'_{i,j}$ for a

fixed input from RF and adds appropriate noise against $z'_{i,j}$. The advantage of the adversary in the above pseudorandomness game is defined by

$$\mathsf{Adv}^{\mathsf{PR}}_{\mathcal{A}}(\lambda, \delta_1) := |2 \cdot \Pr[b' = b] - 1|.$$

Definition 7. *A PUF has* $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-pseudorandomness *if for any PPT adversary* $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{PR}}_{\mathcal{A}}(\lambda, \delta_1) \leq \epsilon(\lambda)$ *holds.*

Sadeghi et al. [30] assumed an ideal PUF which achieves idealized behavior of PUFs and argued that the ideal PUF must satisfies the same notion. While the ideal PUF assumes no noise (i.e., $\delta_1 = 0$), we carefully defined this notion in a formal way to capture the intrinsic noise observed in real PUFs.

3.7 Tamper-Resilience

One of the motivations to employ a PUF in cryptographic schemes and protocols is to provide resilience to physical attacks at cheaper costs compared to other measures like using a Trusted Platform Module (TPM). Though the existing security models for PUFs do not formally define this property, physical attack against the PUF should not leak any internal structure of the device. We consider the following simulation based definition of tamper-resilience. That is, we consider two parties: an adversary $\mathcal{A}$ and a simulator $\mathcal{S}$. The adversary $\mathcal{A}$ can issue $(\mathsf{Create}, \mathsf{Response})$ queries as in the previous definitions. Moreover, whenever $\mathsf{Create}(param)$ is launched, $\mathcal{A}$ receives the produced PUF f_i and can analyze it physically. That is, $\mathcal{A}$ can mount arbitrary physical attacks on the PUF (e.g., power analysis, probing attack, etc). On the other hand, the algorithm $\mathcal{S}$ can only adaptively issue $(\mathsf{Create}, \mathsf{Response})$ but does not get physical access to the created PUFs. Both of them finally output internal state st. The idea is that if for any adversary $\mathcal{A}$ who has physical access to a PUF, there exists a simulator $\mathcal{S}$ which behaves practically the same but without physical access, then the consequence is that the physical access does not provide any advantage. In this case, we say that the PUF is tamper resilient. The advantage of $\mathcal{A}$ in the above experiment is defined by

$$\mathsf{Adv}^{\mathsf{Tamp}}_{\mathcal{A},\mathcal{S},\mathcal{B}}(\lambda) := \left| \begin{array}{l} \Pr[\mathcal{B}(1^\lambda, st) \to 1 \,|\, st \leftarrow \mathcal{A}^{\mathsf{Create},\mathsf{Response}}(1^\lambda, \mathcal{MP}, param, f_1, f_2, \ldots)] \\ \quad - \Pr[\mathcal{B}(1^\lambda, st) \to 1 \,|\, st \leftarrow \mathcal{S}^{\mathsf{Create},\mathsf{Response}}(1^\lambda, \mathcal{MP}, param)] \end{array} \right|$$

where $\mathcal{B}$ is a distinguisher who tries to distinguish st generated by $\mathcal{A}/\mathcal{S}$.

Definition 8. *A function* f *is a* $(\mathcal{MP}, n, \ell, \epsilon)$*-tamper resilient PUF if for any PPT adversary* $\mathcal{A}$*, there exists a PPT algorithm* $\mathcal{S}$*, for any PPT distinguisher* $\mathcal{B}$*,* $\mathsf{Adv}^{\mathsf{Tamp}}_{\mathcal{A},\mathcal{S},\mathcal{B}}(\lambda) \leq \epsilon(\lambda)$ *holds.*

As explained above, the intuition is that the adversary $\mathcal{A}$ actually receives PUFs themselves and hence can conduct different actions in principle, e.g., see the structure of the chip and gate-delay, and launch arbitrary side-channel analysis

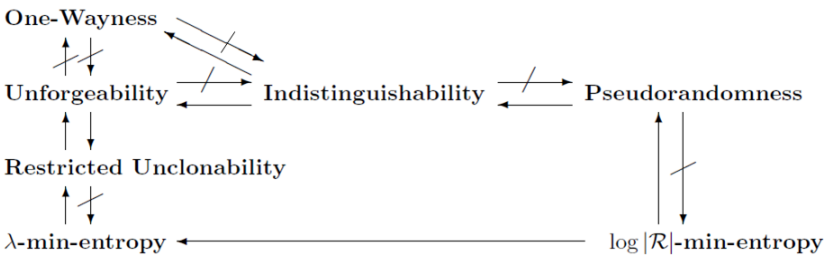

Fig. 1. Relationship among the security properties and min-entropy. For simplicity, we exclude several parameters corresponding to the number of devices, oracle queries and negligible fractions except the amount of min-entropy.

[5]. These results can be contained in st and $\mathcal{B}$ tries to distinguish whether st is output from $\mathcal{A}$ or $\mathcal{S}$. Therefore, if $\mathcal{B}$ cannot distinguish $\mathcal{A}$'s output and $\mathcal{S}$'s output, this means that no additional information which is not trivially derived from challenge-response pairs is extracted by the physical attack (regardless of what they are).

3.8 Relationships between the Security Properties

While each of the security properties had its own separate motivation, we show in the following that these are not completely independent. More precisely, we point out several relationships between these and show the following statements as described in Figure 1 (full formal security proofs are in Appendix B):

- Restricted unclonability is equivalent to EUF-CMA security
- Indistinguishability implies EUF-CMA security and one-wayness
- No implication between one-wayness and EUF-CMA security
- Pseudorandomness implies indistinguishability
- $(\mathcal{MP}, n, |\mathcal{R}|, \lambda, \epsilon)$-min-entropy implies $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-EUF-CMA security
- $(\mathcal{MP}, n, |\mathcal{R}|, \log|\mathcal{R}|, \epsilon)$-min-entropy implies $(\mathcal{MP}, n, \ell, \epsilon)$-pseudorandomness.

4 Comparison to Existing Security Models

[5] We do not limit the number of physical attacks the adversary can mount as defined in [18]. Instead, the pamter-resilience assures there is no extra information is leaked by the physical attacks.

[2] They do not formally define the intra-distance but their implementation results or arguments implicitly show the intra-distance.

[3] Their definition is not information-theoretical min-entropy but computational version of min-entropy called HILL entropy [16].

Table 2. Comparison of output distribution defined in the security models

	Intra-distance	Inter-distance I	Inter-distance II	Min-entropy	Number of PUFs	Number of queries
Pappu [27]	Yes[2]	-	-	-	1	1
Gassend et al. [13]	Yes	Yes	-	-	1	poly
Guajardo et al. [14]	Yes[2]	-	-	-	1	1
Armknecht et al. [2]	Yes	-	-	Yes[3]	1	poly
Armknecht et al. [1]	Yes	-	-	Yes	poly	poly
Brzuska et al. [7]	Yes	-	-	Yes	1	poly
Maes [21]	Yes	-	Yes	-	1	poly
Ours	Yes	Yes	Yes	Yes	poly	poly

Table 3. Comparison of security properties proposed in the security models

	one-wayness	Unforge-ability	Unclon-ability	Indistin-guishability	Pseudo-randomness	Tamper-resilience	Evaluation
Pappu [27]	Yes	UUF-KOA	-	-	-	-	Equality
Gassend et al. [13]	-	UUF-KMA	-	-	-	-	Equality
Guajardo et al. [14]	-	UUF-OT-KMA	-	-	-	-	Equality
Armknecht et al. [2]	-	-	Yes	-	-	-	-
Armknecht et al. [1]	-	UUF-KMA EUF-CMA	Yes	-	-	-	Equality[4] Equality
Brzuska et al. [7]	Yes	EUF-CMA	Yes	-	-	-[5]	Equality
Maes [21]	Yes	UUF-CMA	Yes	-	-	-	Distance
Ours	Yes	EUF-CMA	Yes	Yes	Yes	Yes	Distance

In this section, we compare our model to previous models [1, 2, 7, 13, 14, 21, 27]. An overview is given in Tables 2 and 3. We provide the prior security definitions in Appendix C and discuss here only the differences with our definition.

In all previous models PUF outputs are noisy and hence they consider their intra-distance of outputs. However, the two metrics of inter-distance which refer to evaluations on either multiple inputs or multiple devices are not comprehensively discussed but have been considered in [13] and [21], respectively. This is somewhat surprising, since if the intra-distance is not smaller than the two inter-distances (see discussion in Section 3.1), many security properties are trivially broken (including the unforgeability defined in each paper). In fact, the notions of intra-distance and inter-distance are widely known to implementation designers, but have not been formally captured, e.g., see [14, 19, 20, 23].

As one can see from Table 3 (and as discussed in Section 1), our model covers more security properties than the previous models. This flexibility allows us to express more combinations of different security properties which, in turn, is advantageous for protocol designers to capture needed underlying security assumptions. A further difference is that previous work hardly discussed the

relation between different security properties (and if, then often only in a heuristic sense, e.g., [7]) while, for reasoning about realization, it is crucial to prove which notion is stronger/weaker than another.

Another advantage of our definitions from a theoretical view point is that the intrinsic noise caused by the device is accurately reflected in the definition of an adversary's advantage. It is well known for implementation designers that PUFs output noisy data, and further how to efficiently derive a random but fixed output with a fuzzy extractor or other techniques; see [4, 17, 23]. On the other hand, the previous security models except [1] do not cover the noise in evaluating the advantage of the adversary in their security properties. Estimating the exact noise is intractable and their models cannot fairly evaluate the adversarial advantage. We argue that this neglects an important aspect of PUFs. For example, the higher the noise in the output distribution, the more likely it gets that two PUFs show indistinguishable behavior, and the easier it may become to create clones. Similar thoughts regarding noise apply to almost all security properties. Of course, one possible solution to the above specific issue would be to consider not the PUF alone but only in combination with an appropriate fuzzy extractor as in [1]. However, this approach does not capture the actual requirements for the PUF itself and may fail to cover cases where a PUF is not combined with a fuzzy extractor. Apart from this, a cryptographic protocol may require dedicated security properties of the deployed fuzzy extractor, e.g., see [6]. Hence, we think that the security for PUFs should be argued separately from its typically adjoined building blocks.

Somewhat surprising, we observed that even a seemingly straightforward notion of unforgeability has been treated differently in existing literature. To highlight these differences, we express them using the canonical terminology used for digital signatures and MACs (see Table 3). We specifically stress that our definition of unforgeability covers a stronger attack model compared to other models, since we allow the adversary to obtain direct PUF responses from multiple devices and oracle queries.

Finally, we want to point to the work of Delvaux et al. [10] where different security aspects of PUF-based protocols are discussed. Since their work does not treat security properties for PUFs formally, we do not compare our security model with their informal arguments.

5 Conclusion

In this paper, we proposed a new extended security model for PUFs motivated by existing models, typical demands of cryptographic protocols, but also based on our own considerations about the nature of PUFs. Compared to the existing works, our model is more comprehensive, and presents security definitions that

[4] As we noted in Section 3.3, this model concentrates on a combination of PUF and fuzzy extractor and the evaluation with equality is a natural result.

[5] They argue the necessity of the tamper-resilience in the full version of [7], but no formal definition is described.

are either new or stronger, (e.g., by allowing an adversary to query multiple devices). We also extended these definitions by taking PUF output distributions directly into account.

Formalizing security definitions with multiple properties, first, helps protocol designers to extract the actual requirements for PUF constructions, and, secondly, helps implementation designers to easily find which security properties the proposed PUF construction possesses. Moreover, having a unified security model allows to compare the security of different PUFs and different PUF models. We see our model as a significant step towards this goal.

In light of our methodology, various open questions remain. For example: Are all relevant security properties included in the model or are some missing? Furthermore, due to the physical nature of PUFs, it is often difficult to assess given a concrete PUF, if and what security properties are met. Thus, for the sake of applicability, a PUF security model should allow an engineer to evaluate for a PUF whether certain properties are fulfilled (at least to some extent). While our model follows common cryptographic considerations and models, one cannot rule out that adaptations of the definitions (within our methodology) would make them more applicable for engineers. This clear interdisciplinary task, is a natural open question.

References

1. Armknecht, F., Maes, R., Sadeghi, A., Standaert, F., Wachsmann, C.: A formalization of the security features of physical functions. In: IEEE S&P 2011. pp. 397–412. IEEE Computer Society (2011)
2. Armknecht, F., Maes, R., Sadeghi, A., Sunar, B., Tuyls, P.: Memory leakage-resilient encryption based on physically unclonable functions. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 685–702. Springer, Heidelberg (2009)
3. Armknecht, F., Moriyama, D., Sadeghi, A., Yung, M.: Towards a unified security model for physically unclonable functions. In: CT-RSA 2016. pp. xx – xx, to appear (2016)
4. Bösch, C., Guajardo, J., Sadeghi, A., Shokrollahi, J., Tuyls, P.: Efficient helper data key extractor on FPGAs. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 181–197. Springer, Heidelberg (2008)
5. Boureanu, I., Ohkubo, M., Vaudenay, S.: The limits of composable crypto with transferable setup devices. In: Bao, F., Miller, S., Zhou, J., Ahn, G. (eds.) ASIACCS 2015. pp. 381–392. ACM (2015)
6. Boyen, X.: Reusable cryptographic fuzzy extractors. In: Atluri, V., Pfitzmann, B., McDaniel, P.D. (eds.) ACMCCS 2004. pp. 82–91. ACM (2004)
7. Brzuska, C., Fischlin, M., Schröder, H., Katzenbeisser, S.: Physically uncloneable functions in the universal composition framework. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 51–70. Springer, Heidelberg (2011), full version is available at ePrint Archive 2011/681.
8. Busch, H., Katzenbeisser, S., Baecher, P.: PUF-based authentication protocols - revisited. In: Youm, H.Y., Yung, M. (eds.) WISA 2009. LNCS, vol. 5932, pp. 296–308. Springer, Heidelberg (2009)

9. Dachman-Soled, D., Fleischhacker, N., Katz, J., Lysyanskaya, A., Schröder, D.: Feasibility and infeasibility of secure computation with malicious pufs. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 405–420. Springer, Heidelberg (2014)
10. Delvaux, J., Gu, D., Peeters, R., Verbauwhede, I.: A survey on lightweight entity authentication with strong PUFs. IACR Cryptology ePrint Archive p. 977 (2014)
11. van Dijk, M., Rührmair, U.: Physical unclonable functions in cryptographic protocols: Security proofs and impossibility results. Cryptology ePrint Archive, Report 2012/228 (2012)
12. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM J. Comput. 38(1), 97–139 (2008)
13. Gassend, B., Clarke, D.E., van Dijk, M., Devadas, S.: Silicon physical random functions. In: Atluri, V. (ed.) ACMCCS 2002. pp. 148–160. ACM (2002)
14. Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: FPGA intrinsic PUFs and their use for IP protection. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 63–80. Springer, Heidelberg (2007)
15. Hammouri, G., Sunar, B.: PUF-HB: A tamper-resilient HB based authentication protocol. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 346–365. Springer, Heidelberg (2008)
16. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. 28(4), 1364–1396 (1999)
17. Hofer, M., Böhm, C.: An alternative to error correction for SRAM-like PUFs. In: Mangard, S., Standaert, F. (eds.) CHES 2010. LNCS, vol. 6225, pp. 335–350. Springer, Heidelberg (2010)
18. Kardas, S., Celik, S., Bingöl, M.A., Kiraz, M.S., Demirci, H., Levi, A.: k-strong privacy for radio frequency identification authentication protocols based on physically unclonable functions. Wire;. Commun. Mob. Comput 15, 2150–2166 (2013)
19. Katzenbeisser, S., Koçabas, Ü., Rozic, V., Sadeghi, A., Verbauwhede, I., Wachsmann, C.: PUFs: Myth, fact or busted? A security evaluation of physically unclonable functions (PUFs) cast in silicon. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 283–301. Springer, Heidelberg (2012)
20. Krishna, A.R., Narasimhan, S., Wang, X., Bhunia, S.: MECCA: A robust low-overhead PUF using embedded memory array. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 407–420. Springer, Heidelberg (2011)
21. Maes, R.: Physically Unclonable Functions - Constructions, Properties and Applications. Springer (2013)
22. Maes, R., Herrewege, A.V., Verbauwhede, I.: PUFKY: A fully functional PUF-based cryptographic key generator. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 302–319. Springer, Heidelberg (2012)
23. Maes, R., Tuyls, P., Verbauwhede, I.: Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 332–347. Springer, Heidelberg (2009)
24. Majzoobi, M., Rostami, M., Koushanfar, F., Wallach, D.S., Devadas, S.: Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching. In: IEEE S&P 2012. pp. 33–44. IEEE Computer Society (2012)
25. Oren, Y., Sadeghi, A., Wachsmann, C.: On the effectiveness of the remanence decay side-channel to clone memory-based PUFs. In: Bertoni, G., Coron, J. (eds.) CHES 2013. LNCS, vol. 8086, pp. 107–125. Springer, Heidelberg (2013)

26. Ostrovsky, R., Scafuro, A., Visconti, I., Wadia, A.: Universally composable secure computation with (malicious) physically uncloneable functions. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 702–718. Springer, Heidelberg (2013)
27. Pappu, R.: Physical one-way functions. In: PhD Thesis. MIT (2001)
28. Rührmair, U., van Dijk, M.: Pufs in security protocols: Attack models and security evaluations. In: IEEE S&P 2013. pp. 286–300. IEEE Computer Society (2013)
29. Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., Schmidhuber, J.: Modeling attacks on physical unclonable functions. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACMCCS 2010. pp. 237–249. ACM (2010)
30. Sadeghi, A., Visconti, I., Wachsmann, C.: PUF-enhanced RFID security and privacy. In: SECSI 2010. pp. 366–382 (2010)
31. Saha, I., Jeldi, R.R., Chakraborty, R.S.: Model building attacks on physically unclonable functions using genetic programming. In: HOST 2013. pp. 41–44. IEEE Computer Society (2013)
32. Tuyls, P., Skoric, B.: Strong authentication with physical unclonable functions. In: Security, Privacy, and Trust in Modern Data Management, pp. 133–148. Springer, Heidelberg (2007)

A On the Adversarial Advantage in the One-wayness

Recall that we defined the advantage of the adversary as $\mathsf{Adv}^{\mathsf{OW}}_{\mathcal{A}}(\lambda, \delta_1) := \Pr[y^* = y_1^*] - (\ell+1)/|\mathcal{D}|$ in Section 3.2. We intentionally omit to consider the absolute value of $\mathsf{Adv}^{\mathsf{OW}}_{\mathcal{A}}(\lambda, \delta_1)$ to capture the situation $D \subseteq \mathsf{poly}$. In this case, even if $\mathcal{A}$ does not issue $\mathsf{Response}$, $\mathcal{A}$ can guess y^* with probability $1/|\mathcal{D}|$. Because $\mathcal{A}$ can adaptively issue $\mathsf{Response}$, each query gives a chance for the adversary to guess y^* with probability $1/|\mathcal{D}|$. Therefore, we remove the absolute value and the adversarial advantage is proportionally reduced by ℓ. In the case of typical one-way function, $\mathcal{D}$ is assumed as exponentially large and $\ell = |\mathcal{D}|$ can be a negligible fraction. However, in the case of PUF, the size of $\mathcal{D}$ depends on the concrete structure of PUF (e.g., the domain space of memory based PUFs is actually polynomial).

B Security proofs for Relationship among the Security Definitions

Theorem 1. *The $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-EUF-CMA security is equivalent to the $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-restricted unclonability when $\mathcal{D} \subseteq \mathsf{poly}$.*

Lemma 1. *The $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-EUF-CMA security implies the $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-restricted unclonability.*

Proof. If an adversary $\mathcal{A}$ can break the restricted unclonability of the function f, we show there exists an adversary $\mathcal{B}$ who breaks the EUF-CMA security. Upon receiving $(1^\lambda, \mathcal{MP}, , param)$ from a challenger against the EUF-CMA security, $\mathcal{B}$ sends them to $\mathcal{A}$ and simulates the unclonability game. When $\mathcal{A}$ issues the oracle query with input $(param', i, y_j)$, $\mathcal{B}$ directly transfers it to the challenger and

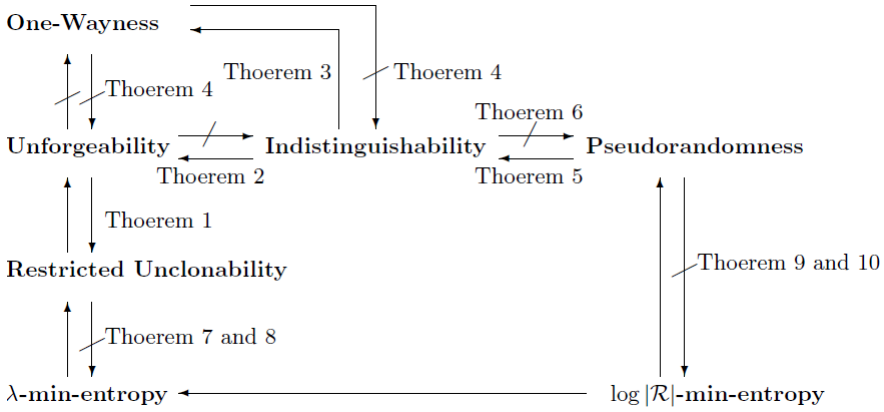

Fig. 2. Relationship among the security properties and corresponding theorems

responds the output. When $\mathcal{A}$ submits $(param, i^*, param', j^*)$, $\mathcal{B}$ selects $y^* \in \mathcal{D}$ which has not been issued as $\mathsf{Response}(param, i^*, y^*)$ and issues $(param', j^*, y^*)$. Upon receiving z^* from a challenger, $\mathcal{B}$ outputs (i^*, y^*, z^*) and finishes the experiment.

Since $\mathcal{A}$ violates the unclonability and successfully finds a function f'_{j^*} created from $param'$ which behaves as f_{i^*}, $\mathsf{Dist}(f'_{j^*}(y^*), z^*) = \mathsf{Dist}(f_{i^*}(y^*), z^*) \leq \delta_1$ and $\mathcal{B}$ can win the EUF-CMA security game. Note that we have at least one $y^* \in \mathcal{D}$ which has not been issued of the form $\mathsf{Response}(param, i^*, y^*)$ from the limitation denoted in the restricted unclonability game. Therefore, EUF-CMA security implies the restricted unclonability.

Lemma 2. *The $\mathcal{MP}, (n, \ell, \delta_1, \epsilon)$-restricted unclonability implies the $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-EUF-CMA security when $\mathcal{D} \subseteq \mathsf{poly}$.*

Proof. If an adversary $\mathcal{A}$ can break the EUF-CMA security of the function f, we show there exists an adversary $\mathcal{B}$ who breaks the unclonability. Similar to the previous security proof, $\mathcal{B}$ internally runs $\mathcal{A}$ and simulates the EUF-CMA security game. Upon receiving $\mathsf{Response}(param', i, y_j)$ from $\mathcal{A}$, $\mathcal{B}$ sends it to the challenger and responds $z_{i,j}$. When $\mathcal{A}$ finally submits (i^*, y^*, z^*), $\mathcal{A}$ performs the following.

1. Set $\mathcal{D}' = \mathcal{D} \setminus \{y^*, \mathcal{Y}\}$ where $\mathcal{Y} := \{y_j\}_j$ contains all input messages $\mathcal{B}$ issues the challenger of the form $\mathsf{Response}(param, i^*, y_j)$. Let $z_{i^*,j} = f(x_{i^*}, y_j)$ be the output derived from the challenger.
2. Send $(param, i^*, y_{j'})$ to the challenger (against the unforgeability game) and obtain $z_{i^*,j'}$ for any $y_{j'} \in \mathcal{D}'$.

3. Output $\mathsf{Create}(param^*)$ where $param^*$ specifies the following parameter to create a new PUF.

$$\begin{cases} f'_{j^*}(y^*) = z^* \\ \forall y_j \in \mathcal{Y}, f'_{j^*}(y_j) = z_{i^*,j} \\ \forall y_{j'} \in \mathcal{D}', f'_{j^*}(y_{j'}) = z_{i^*,j'} \end{cases}$$

4. Upon receiving j^*, putput $(param, i^*, param*, j^*)$.

It is clear that f'_{j^*} performs as f_{i^*} for any input in $\mathcal{R}$. Though almost all input-output pairs are derived from the oracle query, $\mathcal{B}$ does not issue $\mathsf{Response}(param, i^*, y^*)$ to the challenger and its corresponding output z^* is given from $\mathcal{A}$. Therefore, $\mathcal{B}$ can breaks the notion of restricted unforgeability.

Remark that the above proof strategy is valid since we now assume $\mathcal{D} \subseteq \mathsf{poly}$. If $\mathcal{D}$ is larger than a polynomial size, $\mathcal{B}$ cannot issue the oracle query to construct $\mathcal{MP}^*$ since ℓ is polynomially bounded.

Theorem 2. *If f is $(\mathcal{MP}, n, \ell, \epsilon)$-indistinguishable PUF, f is also $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-EUF-CMA secure PUF.*

Proof. Let $\mathcal{A}$ be an adversary against the EUF-CMA security game and $\mathcal{B}$ be an adversary against the indistinguishability game, respectively. We show $\mathcal{B}$ can break the indistinguishability game if the EUF-CMA security is violated by $\mathcal{A}$. $\mathcal{B}$ receives $(1^\lambda, \mathcal{MP}, param)$ from the challenger and internally invokes $\mathcal{A}$. When $\mathcal{A}$ issues $\mathsf{Response}(param', i, y_j)$ as the learning phase of the EUF-CMA security game, $\mathcal{B}$ honestly transfers them to the challenger and responds the result. When $\mathcal{A}$ outputs (i^*, y^*, z^*) as a forgery, $\mathcal{B}$ randomly selects $y_1^* \xleftarrow{\mathsf{U}} \mathcal{D}$ on the condition that $\mathsf{Response}(param, i^*, y_1^*)$ has not been issued to the oracle query. Then (i^*, y^*) and (i^*, y_1^*) is sent to the challenger to proceed the indistinguishability game to the challenge phase. Upon receiving z_b^* from the challenger, $\mathcal{A}$ proceeds the guess phase and outputs 0 if $\mathsf{Dist}(z_b^*, z^*) \leq \delta_1$. Otherwise, $\mathcal{A}$ outputs 1 to the challenger.

When (i^*, y^*, z^*) describes a valid forgery against the PUF, the distance between z^* and $f_{i^*}(y^*)$ is smaller than δ_1. Since this holds iff $b = 0$ in the above indistinguishability game, $\mathcal{B}$ can check and correctly guess the flipped bit b.

Theorem 3. *If f is $(\mathcal{MP}, n, \ell, \epsilon)$-indistinguishable PUF, f is also $(\mathcal{MP}, n, \ell, \delta_1, \epsilon/|\mathcal{D}|^2)$-one-wayness PUF.*

Proof. The proof of this theorem is similar to the proof for Theorem 2. Assume $\mathcal{A}$ is an adversary against one-wayness and $\mathcal{B}$ is an adversary against indistinguishability. $\mathcal{B}$ runs $\mathcal{A}$ internally and simulates the one-wayness game. When $\mathcal{A}$ proceeds to the challenge phase and outputs i^*, $\mathcal{B}$ chooses y_0^*, y_1^* and sends (i^*, y_0^*) and (i^*, y_1^*) to the challenger and receives z_b^*. $\mathcal{B}$ transfers z_b^* to $\mathcal{A}$ and continuously answers to the request of oracle query issued by $\mathcal{A}$. When $\mathcal{A}$ outputs y^* as a guess, $\mathcal{B}$ halts the simulation and outputs b' where $y^* = y_{b'}^*$.

If $\mathcal{A}$ violates the security of one-wayness, $\mathcal{A}$ has an ability to recover y_b^* from $z_b^* \leftarrow f_{i^*}(y_b^*)$. The oracle query issued by $\mathcal{A}$ is correctly answered by $\mathcal{B}$ via oracle query to the challenger (against the indistinguishability game), except the case that y_0^* or y_1^* is issued by $\mathcal{A}$ to the oracle query. Since these inputs are randomly chosen by $\mathcal{B}$, the probability that $\mathcal{B}$ can correctly guess y_0^*, y_1^* which is not issued by $\mathcal{A}$ as $\mathsf{Response}(param, i^*, \cdot)$ is at least $1/|\mathcal{D}|^2$ even if $\mathcal{D} \subset \mathsf{poly}$. We note that $\ell < |\mathcal{D}| - 1$ is necessary for $\mathcal{A}$ to break the security of one-wayness and there are at least two messages in $\mathcal{D}$ which has not been issued to the oracle query. When $\mathcal{D} \supset \mathsf{poly}$ holds, this problem does not happen.

Theorem 4. *There is no implication between one-wayness and unforgeability for PUFs.*

Proof. We consider two simple (theoretical) instantiations of PUFs to show the gap between them. Let f be a PUF which performs a traditional one-way function OWF — e.g., $z_{i,j} := f_i(y_j) = \mathsf{OWF}(i \| y_j)$. It is trivial that this function does not satisfy the unforgeability since anyone can compute OWF with input arbitrary input. On the other hand, the security of OWF prevents any adversary to recover y_j from $z_{i,j}$ so the one-wayness properties is still preserved in f. Thus one-wayness does not imply unforgeability.

Another instantiation of PUF f is $z_{i,j} \leftarrow f_i(y_j) = y_j \| f_{i'}(y_j)$ such that the output of the PUF is just concatenation of the input message y_j and another function $f_{i'}$ which satisfies the unforgeability. The one-wayness property is clearly broken since input y_j is observed from $z_{i,j}$, but f keeps the unforgeability since it is hard to derive a valid forgery against $f_{i'}(y_j)$. Therefore, unforgeability does not imply one-wayness.

We note that the above instantiations also show that both one-wayness and unforgeability do not imply indistinguishability (indistinguishability does not hold for these instantiations).

Theorem 5. *If a PUF f holds $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-pseudorandom, f also satisfies $(\mathcal{MP}, n, \ell, 2\epsilon)$-indistinguishability.*

Proof. Let $\mathcal{A}$ be an adversary against indistinguishability game and $\mathcal{B}$ be an adversary against pseudnrandomness experiment. $\mathcal{B}$ runs $\mathcal{A}$ internally and simulates the oracle queries. Whenever $\mathcal{A}$ asks $\mathsf{Response}(param, i, y_j)$ as Phase 1 or Phase 2, $\mathcal{B}$ asks the same query to the challenger and transfers the response to $\mathcal{A}$. When $\mathcal{A}$ sends $(i_0^*, y_0^*), (i_1^*, y_1^*)$ for the challenge phase, $\mathcal{B}$ selects $b' \xleftarrow{\mathsf{U}} \{0,1\}$ and asks $\mathsf{Response}(param, i_{b'}^*, y_{b'}^*)$ to the challenger. Depend on the flippled coin $b \xleftarrow{\mathsf{U}} \{0,1\}$, the challenger responds z^* to $\mathcal{B}$ and it is also forwarded to $\mathcal{A}$ by $\mathcal{B}$. When $\mathcal{A}$ finishes the game and outputs b'', $\mathcal{B}$ outputs the same bit and halts the simulation.

Consider the case that $\mathcal{B}$ selects $b' = 0$. Then z^* is determined by the coin b flipped by the challenger in the pseudorandom game. If $\mathcal{A}$ distinguishes the difference and the output b'' is changed for b, this means $\mathcal{B}$ can break the security of pseudorandomness. The same argument is also applied for $b' = 1$. ON the other

hand, if we concentrate on the situation that $b = 1$, it is impossible for $\mathcal{A}$ to estimate b' from z^* since z^* is only selected from RF and a small noise is added. That is, z^* is completely independent from b'. Thus, if f is $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-pseudorandom function, f is also $(\mathcal{MP}, n, \ell, 2\epsilon)$-indistinguishability.

Theorem 6. *$(\mathcal{MP}, n, \ell, \epsilon)$-indistinguishability does not imply $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-pseudorandomness.*

Proof. Assume a function f satisfies $(\mathcal{MP}, n, \ell, \epsilon)$-indistinguishability. Consider another function $f' : \mathcal{D} \to \{0,1\}^\lambda \times \mathcal{R}$ which takes as input $y \in \mathcal{D}$ and outputs $1^\lambda \| f(y) \in \{0,1\}^\lambda \times \mathcal{R}$. That is, the output of f' contains a sufficiently large prefix for any inputs. It is clear that f' cannot satisfy pseudorandomness when an adversary can count the number of 1's in the λ-bit prefix of an output. On the other hand, f still preserves indistinguishability since a prefix is applied for any input and useless to increase the advantage (the remaining output sequence from f holds indistinguishability). Thus $(\mathcal{MP}, n, \ell, \epsilon)$-indistinguishability does not imply $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-pseudorandomness.

Theorem 7. *$(\mathcal{MP}, n, |\mathcal{R}|, \lambda, \epsilon)$-min-entropy implies $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-EUF-CMA security.*

Proof. Assume that there is an adversary who breaks the security of EUF-CMA with non-negligible probability larger than $\epsilon := 2^{-\lambda}$. Then we have

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{EUF\text{-}CMA}}(\lambda, \delta_1) = \Pr[\mathsf{Dist}(z^*, f_{i^*}(y^*)) \leq \delta_1] - \frac{|\mathcal{Z}'|}{|\mathcal{R}|} > 2^{-\lambda}.$$

Because $\frac{|\mathcal{Z}'|}{|\mathcal{R}|}$ denotes the randomly guessing probability, we remove this affect and obtain $\Pr[z^* = f_{i^*}(y^*)] > 2^{-\lambda}$. This contradicts to the $(n, |\mathcal{R}|, \lambda, \epsilon)$-min-entropy which asserts any output has min-entropy λ and its estimation is limited by $2^{-\lambda}$. Therefore, we obtain that $(n, |\mathcal{R}|, \lambda, \epsilon)$-min-entropy implies $(n, \ell, \delta_1, \epsilon)$-EUF-CMA security.

The reason why we require the min-entropy with $\ell := |\mathcal{R}|$ is that the adversary against the EUF-CMA security can issue the oracle query and obtain arbitrary input-output pair of the PUF. Therefore, if we concentrate on the EUF-KMA security, we can easily say that $(\mathcal{MP}, n, \ell, \lambda, \epsilon)$-min-entropy implies $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-EUF-KMA for arbitrary $\ell \in \{1, \ldots, |\mathcal{R}|\}$.

Theorem 8. *$(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-EUF-CMA security does not imply $(\mathcal{MP}, n, |\mathcal{R}|, \lambda, \epsilon)$-min-entropy.*

Proof. Even $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-EUF-CMA security holds, this notion is only valid for computational (probabilistic polynomial time) adversary. Because the traditional min-entropy implicitly requires an information-theoretical adversary, we cannot simply say that $(\mathcal{MP}, n, \ell, \delta_1, \epsilon)$-EUF-CMA security implies $(\mathcal{MP}, n, |\mathcal{R}|, \lambda, \epsilon)$-min-entropy. One simple example is $f_i(y_j) := \mathsf{MAC}(sk, i \| y_j)$ where MAC denotes a message authentication code and $sk \xleftarrow{\mathsf{U}} \{0,1\}^\lambda$ is a fixed random

string. It is clear that the PUF holds EUF-CMA security if the MAC function is securely chosen. On the other hand, the above function is deterministic and $\tilde{H}_\infty(sk \mid f_i(y_1), f_i(y_2)) = \tilde{H}_\infty(f_i(y_1) \mid f_i(y_2)) = 0$ for any $y_1, y_2 \in \mathcal{R}$ in an information theoretical setting.

Theorem 9. *$(\mathcal{MP}, n, |\mathcal{R}|, \log|\mathcal{R}|, \epsilon)$-min-entropy implies $(\mathcal{MP}, n, \ell, \epsilon)$-pseudorandomness.*

Proof. This is clear that $(n, \ell, \log|\mathcal{R}|, \epsilon)$-min-entropy implies that the PUF's output has a full entropy and there is no biased bit and correlation with the other outputs. Therefore the PUF's output from f is indistinguishable from randomized selection as described in the pseudorandomness. The converse is trivial because any outputs from a random function have full-entropy (e.g., $\log|\mathcal{R}|$-bit min-entropy) and δ_1-bit randomly chosen noise does not decrease the entropy.

Theorem 10. *$(\mathcal{MP}, n, \ell, \epsilon)$-pseudorandomness does not imply $(\mathcal{MP}, n, \ell, \log|\mathcal{R}|, \epsilon)$-min-entropy.*

Proof. The difference from the computational complexity and information theory provides the gap between these notions analogous to the proof of Theorem 8. Let $f_i(y_j) := \mathsf{PRF}(sk, i, y_j)$ and $\delta_1 = 0$ where PRF is a secure pseudorandom function and sk is a random seed for the pseudorandom function. It is obvious that f holds (n, ℓ, ϵ)-pseudorandomness (note that $\delta_1 = 0$ means typical pseudorandomness). On the other hand, the output is deterministically defined by the input and f provides no additional min-entropy if we consider the conditional entropy. Analogous as the proof for Theorem 8, we have $\tilde{H}_\infty(sk \mid f_i(y_1), f_i(y_2)) = \tilde{H}_\infty(f_i(y_1) \mid f_i(y_2)) = 0$ for any $y_1, y_2 \in \mathcal{R}$ in an information theoretical setting.

C Existing Security Models

C.1 Pappu 2001 [27]

The notion of PUF is originally introduced by Pappu [27] and he requires the following issue. A function f is a PUF if for any adversary $\mathcal{A}$,

$$\Pr\left[y' = y \,\middle|\, \begin{array}{c} y \xleftarrow{\cup} \mathcal{D}, z \leftarrow f(y), \\ y \leftarrow \mathcal{A}(1^\lambda, f, z) \end{array}\right] \leq \epsilon(\lambda).$$

The above probability evaluates whether the adversary can estimate the PUF's input for an output z. He gave another issue such that the adversary cannot estimate the output y such that

$$\Pr\left[z' = z \,\middle|\, \begin{array}{c} y \xleftarrow{\cup} \mathcal{D}, z \leftarrow f(y), \\ z' \leftarrow \mathcal{A}(1^\lambda, f, y) \end{array}\right] \leq \epsilon(\lambda).$$

C.2 Gassend et al. 2002 [13]

Gassend et al. give another formulation for PUF [13]. A function f is a PUF if for any adversary $\mathcal{A}$,

$$\Pr\left[z' = z \;\middle|\; \begin{array}{c} y \xleftarrow{\cup} \mathcal{D}, z \leftarrow f(y), \\ \{y_i \xleftarrow{\cup} \mathcal{D}, z_i \leftarrow f(y_i)\}_{i=1,\ldots,\ell}, \\ z' \leftarrow \mathcal{A}(1^\lambda, f, y, \{(y_i, z_i)\}_i) \end{array}\right] \leq \epsilon(\lambda).$$

Different from the requirement in Section C.1, the adversary can learn input-output pairs parameterized by ℓ before estimating the output for the challenge input y.

C.3 Guajardo et al. 2007 [14]

Guajardo et al. introduced the following issue for PUF to be used in a hardware-based authentication protocol for IP protection [14].

1. The adversary cannot estimate the output even when a different input-output pair is given.

$$\Pr\left[z' = z_2 \;\middle|\; \begin{array}{c} x \in \mathcal{K}, y_1, y_2 \xleftarrow{\cup} \mathcal{D} \\ z_1 \leftarrow f(y_1), z_2 \leftarrow f(y_2), \\ z' \leftarrow \mathcal{A}(1^\lambda, f, y_1, z_1, y_2) \end{array}\right] \leq \epsilon(\lambda)$$

2. The adversary cannot estimate a valid input-output pair on the condition that no pair can be observed.

$$\Pr\left[z' = f(y') \;\middle|\; \begin{array}{c} x \in \mathcal{K} \\ (y', z') \leftarrow \mathcal{A}(1^\lambda, f) \end{array}\right] \leq \epsilon(\lambda)$$

3. When the adversary launches a physical attack against the PUF, the PUF is broken, e.g., its behavior becomes independent from the original execution.

C.4 Armknecht et al. 2009 [2]

Armknecht et al. [2] provides the following issue.

1. The Hamming distance (HD) between the two outputs for the same input and fixed device is at most δ. That is,

$$\Pr\left[\mathsf{HD}(z_1, z_2) \leq \delta \;\middle|\; \begin{array}{c} y \xleftarrow{\cup} \mathcal{D}, \\ z_1 \leftarrow f(y), z_2 \leftarrow f(y) \end{array}\right] = 1.$$

2. There exists a distribution $\mathcal{D}$ such that for arbitrary input y, $\{z_i \mid z_i := f_i(y)\}_i$ is indistinguishable from $\{z_i \mid z_i \xleftarrow{\cup} \mathcal{D}\}_i$ except with negligible probability ϵ. They assume $\tilde{H}_\infty(\mathcal{D}) > 0$[6].
3. There is no technology to physically clone the device.
4. If a physical attack is executed, the functionality of the PUF is changed or broken.

[6] When $\tilde{H}_\infty(\mathcal{D}) \geq q$, we can say that the PUF's distribution has HILL entropy at least q. The HILL entropy is the computational version of min-entropy measurement [16].

C.5 Armknecht et al. 2011 [1]

Armknecht et al. [1] described a security model for a PUF-enabled system. They consider a scenario that the output from the PUF is input to the fuzzy extractor. Their composed system is called as PFS (PUF System) and they provide a security model for PFS. Let $\mathsf{FE} = (\mathsf{FE.Gen}, \mathsf{FE.Rec})$ be an algorithm for a fuzzy extractor. $\mathsf{PFS}(y, h)$ outputs $(f(y), \mathsf{FE.Gen}(f(y)))$ if $h = \emptyset$. Otherwise, $\mathsf{PFS}(y, h)$ outputs $(\mathsf{FE.Rec}(f(y), h), h)$.

1. When the PFS is evaluated with the same input and fixed device, the outputs are the same. That is,
$$\Pr\left[z_1 = z_2 \left| \begin{array}{c} \forall y \xleftarrow{\cup} \mathcal{D}, \\ (z_1, h) \leftarrow \mathsf{PFS}(y), z_2 \leftarrow \mathsf{PFS}(y, h) \end{array} \right.\right] = 1.$$
2. There is no chance for an adversary to find two devices which the corresponding outputs are observed for a subset of the domain. When the adversary adaptively submit an index i, a challenger gives the adversary device $f_i(\cdot)$. Even when the adversary submits two PUF systems $\mathsf{PFS}_1^*(\cdot)$ and $\mathsf{PFS}_2^*(\cdot)$, we have
$$\Pr\left[z_1 = z_2 \left| \begin{array}{c} \exists \mathcal{D}' \subseteq \mathcal{D}, \forall y \in \mathcal{D}', \\ (z_1, h) \leftarrow \mathsf{PFS}_1^*(y), z_2 \leftarrow \mathsf{PFS}_2^*(y, h) \end{array} \right.\right] \leq \epsilon(\lambda).$$
3. The adversary cannot estimate the output for a target input with a lot of input-output pairs. For any adversary $\mathcal{A}$,
$$\Pr\left[z' = z \left| \begin{array}{c} y \xleftarrow{\cup} \mathcal{D}, (z, h) \leftarrow \mathsf{PFS}(y), \\ \{y_i \xleftarrow{\cup} \mathcal{D}, (z_i, h_i) \leftarrow \mathsf{PFS}(y_i)\}_{i=1,\ldots,\ell}, \\ z' \leftarrow \mathcal{A}(1^\lambda, f, y, h, \{(y_i, z_i, h_i)\}_i) \end{array} \right.\right] \leq \epsilon(\lambda).$$
4. Even if the adversary adaptively obtains the outputs with oracle queries, no adversary cannot guess a valid input-output pair. This is formalized with the following game between a challenger and adversary $\mathcal{A}$.
   Setup. The challenger gives $(1^\lambda, f)$ to the adversary $\mathcal{A}$.
   Evaluation. When $\mathcal{A}$ sends a pair of index and input (i, y_j, h_j), the challenger responds $(z_{i,j}, h_j) \leftarrow \mathsf{PFS}_i(y_j, h_j)$. The adversary can adaptively issue the oracle.
   Guess. When the adversary finishes the evaluation phase, $\mathcal{A}$ outputs (i^*, y^*, z^*, h^*).
   Then, $\Pr[z^* = z \mid (z, h^*) := \mathsf{PFS}_i^*(y^*, h^*)] \leq \epsilon(\lambda)$ holds for any adversary $\mathcal{A}$.

C.6 Brzuska et al. 2011 [7]

Brzuska et al. introduced a universal composability framework for PUFs in [7]. They require that the PUF must hold the following properties:

1. If the PUF is computed twice, the Hamming distance between them is at most δ.

$$\Pr\left[\mathsf{HD}(z_1, z_2) \leq \delta \middle| \begin{array}{c} x \in \mathcal{K}, y \xleftarrow{\cup} \mathcal{D}, \\ z_1 \leftarrow f(y), z_2 \leftarrow f(y) \end{array}\right] = 1.$$

2. When a target input y^* and other inputs $(y_1, \ldots, y_\ell)$ have Hamming distance at least δ', the target output $z^* \leftarrow f(y^*)$ has sufficient min-entropy δ'' on the condition that the results of PUF with input $(y_1, \ldots, y_\ell)$ are observed. That is,

$$\Pr\left[\tilde{H}_\infty(z^* \mid z_1, \ldots, z_\ell) \geq \delta'' \middle| \begin{array}{c} y^*, y_1, \ldots, y_\ell \xleftarrow{\cup} \mathcal{D}, \\ \mathsf{HD}(y^*, y_i) \geq \delta' \ (1 \leq \forall i \leq \ell), \\ z^* \leftarrow f(y^*), \{z_i \leftarrow f(y_i)\}_{1 \leq i \leq \ell} \end{array}\right] = 1.$$

3. PUF satisfies the requirements of the universal composability framework. The intuition in the UC framework is formalize the gap between the interaction with the actual protocol and idealized protocol behavior. The idealized behavior is captured by an description called ideal functionality, and it intermediates communication between the protocol participants and adversary. Brzuska et al. proposed the following ideal functionality $\mathcal{F}_{\mathsf{PUF}}$ for PUFs.

Intuitively, the above ideal functionality defines the ownership of the PUF. Only if a party has the ownership of the PUF ($(\mathtt{sid}, \mathsf{id}, P_i, \mathsf{notrans}) \in L$), ideal functionality responds the output of the PUF with input from the party. During the period between a party requests the ownership transfer and a malicious adversary declares to finish the transmission, the adversary can interact to the PUF with the oracle query.

While this ideal functionality may be a candidate of ideal behavior of the PUFs, we cannot identify this is actually related to concrete security properties to provide provable security in a cryptographic protocol. Thus we do not argue the relationship against the security issues we defined.

C.7 Maes 2013 [21]

Maes provided several properties for PUFs [21, Chapter 3.2]. In his model, each PUF f_i is produced by a creation procedure $f.\mathsf{Create}$. He described the following issues to the PUF.

1. The provability that the noise caused from the two responses derived from the same PUF instance and the same challenge is small is high. Define $\delta_1 := \{\mathsf{Dist}(z_1, z_2) \mid y \in \mathcal{D}, z_1 \leftarrow f(y), z_2 \leftarrow f(y)\}$. Then $\Pr[\delta_1 \text{ is low}]$ is high for any f and y.
2. The provability that the distance between the other output is large is high. Define $\delta_2 := \{\mathsf{Dist}(z_1, z_2) \mid y \in \mathcal{D}, z_1 \leftarrow f_1(y), z_2 \leftarrow f_2(y)\}$. Then $\Pr[\delta_2 \text{ is large}]$ is high for any $f(x, \cdot)$ and y.

$\mathcal{F}_{\mathsf{PUF}}$ interacts with protocol participants $P_1, \ldots, P_n$ and adversary $\mathcal{S}$.

- Upon receiving $(\mathsf{init}_{\mathsf{PUF}}, \mathtt{sid}, P_i)$ from P_i, check a list L (which is initially empty) whether L contains $(\mathtt{sid}, \cdot, \cdot, \cdot)$. If so, $\mathcal{F}_{\mathsf{PUF}}$ ignores the message. Otherwise, record $(\mathtt{sid}, \mathsf{id}, P_i, \mathsf{notrans})$ to L where $\mathsf{id} := f(x, \cdot)$ is randomly chosen from PUF family and send $(\mathsf{initialized}_{\mathsf{PUF}}, \mathtt{sid})$ to P_i.
- Upon receiving $(\mathsf{eval}_{\mathsf{PUF}}, \mathtt{sid}, P_i, y)$ from P_i, check whether $(\mathtt{sid}, \mathsf{id}, P_i, \cdot, \mathsf{notrans})$ is recorded in L. If so, evaluate $z \leftarrow f(y)$ and send $(\mathsf{eval'ed}_{\mathsf{PUF}}, \mathtt{sid}, y, z)$ to P_i. Otherwise, ignore the request.
- Upon receiving $(\mathsf{handover}_{\mathsf{PUF}}, \mathtt{sid}, P_i, P_j)$ from P_i, check whether $(\mathtt{sid}, \cdot, P_i, \mathsf{notrans})$ is recorded in L. If so, update the tuple to $(\mathtt{sid}, \mathsf{id}, \bot, \mathsf{trans}(P_j))$ and send $(\mathsf{invoke}_{\mathsf{PUF}}, \mathtt{sid}, P_i, P_j)$ to $\mathcal{S}$. Otherwise, ignore the request.
- Upon receiving $(\mathsf{eval}_{\mathsf{PUF}}, \mathtt{sid}, \mathcal{S}, y)$ from $\mathcal{S}$, check whether $(\mathtt{sid}, \mathsf{id}, \bot, \mathsf{trans}(P_j))$ is recorded in L. If so, evaluate $z \leftarrow f(x, y)$ and send $(\mathsf{eval'ed}_{\mathsf{PUF}}, \mathtt{sid}, y, z)$ to $\mathcal{S}$. Otherwise, ignore the request.
- Upon receiving $(\mathsf{ready}_{\mathsf{PUF}}, \mathtt{sid}, \mathcal{S})$ from $\mathcal{S}$, check whether $(\mathtt{sid}, \mathsf{id}, \bot, \mathsf{trans}(P_j))$ is recorded in L. If so, update the tuple to $(\mathtt{sid}, \mathsf{id}, P_j, \mathsf{notrans})$, store $(\mathsf{received}_{\mathsf{PUF}}, \mathtt{sid}, P_i)$ and send $(\mathsf{handover}_{sPUF}, \mathtt{sid}, P_i)$ to P_j. Otherwise, ignore the request.
- Upon receiving $(\mathsf{received}_{\mathsf{PUF}}, \mathtt{sid}, P_i)$ from $\mathcal{S}$, check whether this tuple is stored. If so, send this tuple to P_i. Otherwise, ignore the request.

Fig. 3. Ideal functionality $\mathcal{F}_{\mathsf{PUF}}$ proposed by Brzuska et al. [7]

3. There is no efficient invert algorithm to the PUF. For a given $z \leftarrow f(y)$, no algorithm finds y^* such that $\Pr[\mathsf{Dist}(z^*, z) \leq \delta_1 \mid z^* \leftarrow f(y^*)]$ is low.
4. For any influence to the creation procedure $f.\mathsf{Create}$, it is hard to provide two PUFs f_1, f_2 such that $\Pr[\mathsf{Dist}(z_1, z_2) \leq \delta_1 \mid y \in \mathcal{D}, z_1 \leftarrow f_1(y), z_2 \leftarrow f_2(y)]$ is high.
5. Upon receiving multiple responses from a PUF with adaptively chosen inputs and target input y, an adversary tries to estimate the output as z^*. Then we have $\Pr[\mathsf{Dist}(z^*, z) \leq xx \mid z \leftarrow f(y)]$ is low. Depend on the security level, the adversary learns a limited number of input-output pairs (unpredictability) or unbounded numbers of them (mathematical unclonability).
6. Any physical transformation from PUF f_1 to f_2 causes the effect. That is, $\Pr[\mathsf{Dist}(z^*, z) \geq \delta_2 \mid y \in \mathcal{D}, z \leftarrow f_1(y), z^* \leftarrow f_2(y)]$ is high.