

Foundations of Reconfigurable PUFs

Jonas Schneider^(✉) and Dominique Schröder

Saarland University, Saarbrücken, Germany

s9joscne@stud.uni-saarland.de

Abstract. A Physically Unclonable Function (PUF) can be seen as a source of randomness that can be challenged with a stimulus and responds in a way that is to some extent unpredictable. PUFs can be used to provide efficient solutions for common cryptographic primitives such as identification/authentication schemes, key storage, and hardware-entangled cryptography. Moreover, Brzuska et al. have recently shown, that PUFs can be used to construct UC secure protocols (CRYPTO 2011). Most PUF instantiations, however, only provide a static challenge/response space which limits their usefulness for practical instantiations. To overcome this limitation, Katzenbeisser et al. (CHES 2011) introduced Logically Reconfigurable PUFs (LR-PUFs), with the idea to introduce an “update” mechanism that changes the challenge/response behaviour without physically replacing or modifying the hardware.

In this work, we revisit LR-PUFs. We propose several new ways to characterize the unpredictability of LR-PUFs covering a broader class of realistic attacks and examine their relationship to each other. In addition, we reconcile existing constructions with these new characterizations and show that they can withstand stronger adversaries than originally shown. Since previous constructions are insecure with respect to our strongest unpredictability notion, we propose a secure construction which relies on the same assumptions and is almost as efficient as previous solutions.

Keywords: Physically unclonable functions · Logically reconfigurable · Tamper-resistance

1 Introduction

Physically Unclonable Function (PUFs) are non-programmable hardware tokens that can be challenged with a stimulus and output responses that are unpredictable. The unpredictable output of the PUFs results from the manufacturing process and cannot be controlled even by the producer itself. PUFs are extremely useful to build cryptographic applications, such as e.g., identification/authentication schemes, key storage, and hardware-entangled cryptography, and also to obtain protocols that are secure in Canetti’s UC framework as shown by Brzuska et al. [4]. Most PUF instantiations, however, only provide a static challenge/response space which limits their usefulness for practical instantiations. To overcome this limitation, Katzenbeisser et al. [8] introduced Logically

Reconfigurable PUFs (LR-PUFs), with the idea to introduce an “update” mechanism that allows to change the input/output behaviour of a PUF. In this work, we revisit LR-PUFs presenting several ways to characterize the unpredictability, we examine their relationship to each other, and we show that previous constructions can withstand stronger adversaries than originally shown.

1.1 Background and Related Work

Physically Unclonable Functions were proposed as *Physical One-Way Functions* [13]. They consist of a physical device which can be challenged with a stimulus and responds in a way that is to some extent unpredictable.

- The PUF provides unpredictable, but robust responses. This means the response for a given challenge does not vary beyond a typically low bound, but it should not be possible to predict the response for a stimulus that has not yet been applied.
- The PUF is not clonable, i.e., one cannot produce a device which exhibits the same response behavior. This goes even as far as not being able to recreate the same behavior if one has physical access to the device itself and not just a list of challenge-response pairs.

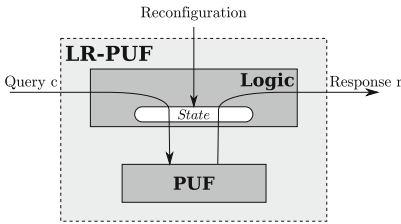


Fig. 1. Schematic of a generic Logically Reconfigurable PUF construction.

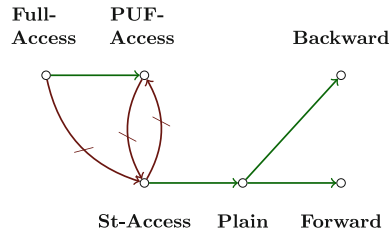


Fig. 2. The relations between the unpredictability notions we introduce.

A third property that is usually cited is tamper-evidence, which is closely related to unclonability. These properties are derived from imprecisions in the manufacturing process of some other object, such as differing gate delays in an integrated circuit. For a survey on the multitude of different PUF constructions, we refer the reader to [11]. A formal description of these properties has been the subject of many research efforts. An in-depth treatment to the definition of these properties that proposes a game-based framework for the description of PUF properties and even PUF creation is given in [2]. Brzuska et al. propose an entropy based characterization of the unpredictability property and examine how PUFs can be integrated into the UC-security framework in [4]. We use their formalization of PUFs as families of distributions.

The tamper-evidence property forbids most PUF designs to have a stimulus-response behavior which is anything but static. In applications where PUFs serve

as physical tokens for e.g., access control this can be a disadvantage. Consider for example that the PUF-token should be transferable to a different person. The traditional PUF designs do not allow this, unless the new owner of the PUF should be allowed to carry the same credentials as the previous owner. Thus, there have been efforts to construct reconfigurable PUFs, the first being Controlled Physical Random Functions [6], and a more recent one being Reconfigurable PUFs, short rPUFs [10].

Both of these approaches have their own limitations. Controlled PUFs effectively limit the PUF response space a single user can draw from, thereby lessening security. Physically reconfigurable rPUFs require a potentially costly physical reconfiguration process, and there are no guarantees regarding the effectiveness of that process.

A solution which aims to sidestep these limitations are Logically Reconfigurable PUFs (LR-PUFs) [8]. In this approach, reconfiguration leaves the physical device untouched and is instead performed on a piece of state, which is stored together with the PUF. The stimulus mechanism of the PUF is encapsulated in a query algorithm, which processes challenges by entangling them in some way with the current state of the device. See Fig. 1 for a visual representation of the LR-PUF concept. The idea is a combination of the state and the challenge to perform a *logical* reconfiguration, in which a new state is chosen instead of altering the actual physical device. This preserves the original input/output characteristics of the PUF, but does not require physical manipulation of the device.

1.2 Applications

PUFs have a wide range of applications such as key extraction and authentication [9, 16–18], remote attestation [16], and tamper-proof and fault injection resilient implementations of cryptographic primitives [1, 3, 15]. Most of these applications assume that the PUFs are somewhat ideal in the sense that they support large challenge and/or response spaces. Since most of the known PUF instantiations do not fulfil these properties, LR-PUFs seem to be better suited. Another application of LR-PUFs are electronic fare systems for public transport as suggested in [8]. In this setting, an access token is equipped with an LR-PUF, that serves to authenticate the passenger at the entry points to the transport system and can be used to secure a credit stored on the token. The reconfiguration capability of the device enables the easy reuse of tokens, as reconfiguration of the LR-PUF is (ideally) equivalent to physically replacing the device without causing the cost of a new device.

Another interesting application of LR-PUF technology is presented in [5]. Here, the LR-PUF is used to provide secure key storage and helps to prevent cloning and downgrading of embedded software authenticated using the stored keys. In this application, software is bound to an embedded device by encrypting it with a device-specific key. This key is generated by querying a PUF that is part of the device, making the key dependent on the unique properties of the PUF in each device. This prevents cloning of the software to a new device, as

the key on a cloned device will be generated differently. The reconfigurability is used in the event of a software update, to prevent downgrading to an older version of the software. In this event, a new key to encrypt the updated software is derived, and the old software version will no longer be useful, because the old key can no longer be retrieved from the LR-PUF.

1.3 Contribution

We revisit the LR-PUFs as introduced in [8] and present several ways to characterize the unpredictability notion. We reconsider existing constructions with respect to these new measures, and we propose a novel construction that is secure w.r.t. our strongest notion of unpredictability. In the following, we discuss each contribution more in detail.

Definitions. We introduce four different notions of unpredictability. The first one is called **Plain**-unpredictability and it is a natural extension of backward/forward-unpredictability of [8]. The basic idea of this definition is to allow the adversary to reconfigure the PUF several times. The second notion, called **St-Access**-unpredictability, removes the assumption that the state is stored in a tamper-evident manner and allows the adversary to directly write the state. The third notion, called **PUF-Access**-unpredictability, models the case where the adversary manages to bypass the query and reconfiguration logic and where it gains direct access to the PUF. The fourth notion, called **Full-Access**-unpredictability, combines **PUF-Access**-unpredictability and **St-Access** unpredictability in the sense that the adversary has direct access to the PUF and is allowed to set the state maliciously. Perhaps surprisingly there is an obstacle when trying to compare the power of state-setting adversaries to PUF-access adversaries. The issue is that a PUF-access adversary might be able to completely precompute the behavior of an LR-PUF given the current state, which makes both notions incomparable. A visual representation of these relations is given in Fig. 2, where an arrow $A \rightarrow B$ denotes, that notion A implies notion B and an arrow $A \not\rightarrow B$ means that notion A does not imply notion B.

Analysis. In Appendix C we give a comprehensive security analysis of the “speed-optimized” and the “area-optimized-construction” from [8] w.r.t. our unpredictability notions. The former employs a collision resistant hash function both to combine state information and query and to generate a new state from the old one. The latter uses an identical reconfiguration algorithm, but is geared towards PUFs with small area, i.e., small input range by providing a query mechanism that involves iteratively constructing a response from smaller sub-queries. (For full definitions, please refer to Fig. 8). Our analysis shows that both constructions are **St-Access**-unpredictable. Previously, it was only known that both constructions are backward (resp. forward) unpredictable. The practical consequences of this result is that the scheme remains secure, even if the state is *not* stored in a tamper-evident manner. On the negative side we show that both constructions are not secure against adversaries that have direct access to

the PUF. In fact, our result here is more general, showing that any LR-PUF cannot satisfy this notion where access to the underlying PUF makes the query and reconfiguration algorithms completely predictable to the adversary.

Construction. We propose a simple LR-PUF construction that is **Full-Access**-unpredictable. Our scheme can be seen as a randomized variant of the “speed-optimized” construction from [8] with the difference being that our reconfiguration algorithm samples a fresh state st upon reconfiguration and it evaluates the underlying PUF on $w \leftarrow \text{Hash}(\text{st} \parallel c)$. This construction relies on the same computational assumptions as the scheme of [8], it is almost as efficient, but it satisfies both **Full-Access**- and **St-Access**-unpredictability.

1.4 Outline

In Sect. 2 we give some background by reviewing a formalization of Physically Unclonable Functions and present our formalization of LR-PUFs. Section 3 introduces the new unpredictability notions we propose and the relations among them. Section 4 contains the specification of a construction which achieves the strongest of our unpredictability notions.

2 Logically Reconfigurable PUFs

2.1 Physically Unclonable Functions

A Physically Unclonable Function (PUF) is a noisy function that is realized through a physical object [13]. The PUF can be queried with a challenge c and answers with a response r . The output of the PUF is noisy meaning that querying the PUF twice with the same challenge yields most likely two different but closely related responses. In the following we recall the definition of PUFs and their main security property given in [4].

Definition 1 (Physically Unclonable Functions). *Let ρ be the dimension of the range of the PUF responses of the PUF family, and let d_{noise} be a bound on the PUF’s noise. A pair $\mathcal{P} = (\mathbf{S}, \mathbf{E})$ is a family of (ρ, d_{noise}) -PUFs if it satisfies the following properties:*

Index Sampling. *Let \mathcal{I}_λ be an index set. The sampling algorithm \mathbf{S} outputs, on input the security parameter 1^λ , an index $\text{id} \in \mathcal{I}_\lambda$. We do not require that the index sampling can be done efficiently. Each index $\text{id} \in \mathcal{I}_\lambda$ corresponds to a set \mathcal{D}_{id} of distributions. For each challenge $c \in \{0, 1\}^\lambda$, \mathcal{D}_{id} contains a distribution $\mathcal{D}_{\text{id}}(c)$ on $\{0, 1\}^{\rho(c)}$. We do not require that \mathcal{D}_{id} has a short description or an efficient sampling algorithm.*

Evaluation. *The evaluation algorithm \mathbf{E} gets as input a tuple $(1^\lambda, \text{id}, c)$, where $c \in \{0, 1\}^\lambda$. It outputs a response $r \in \{0, 1\}^{\rho(\lambda)}$ according to distribution \mathcal{D}_{id} . It is not required that \mathbf{E} is a PPT algorithm.*

Bounded Noise. *For all indices $\text{id} \in \mathcal{I}$, for all challenges $c \in \{0, 1\}^\lambda$, we have that when running $\mathbf{E}(1^\lambda, \text{id}, c)$ twice, then for any two outputs r_1, r_2 that are produced the Hamming distance $\text{dis}(r_1, r_2)$ is smaller than $d_{\text{noise}}(\lambda)$.*

Unpredictability of PUFs. Loosely speaking, a PUF is unpredictable if it is difficult to predict the response of the PUF to a given, previously unknown challenge. This intuition is formalized in an experiment where the adversary can adaptively query the PUF on challenges of its choice and wins if it can predict the response to a fresh challenge of its choice, within the bound d_{noise} . Fresh means that the adversary did not query the PUF on this challenge.

Definition 2 (PUF-Unpredictability). *A family of PUFs $\mathcal{P} = (S, E)$ is unpredictable if for any PPT algorithm \mathcal{A} the probability that the experiment $\text{PRE}_{\mathcal{A}}^{\mathcal{P}}(\lambda)$ evaluates to 1 is negligible (in the security parameter λ), where*

Experiment $\text{PRE}_{\mathcal{A}}^{\mathcal{P}}(\lambda)$

$\text{id} \leftarrow S(1^\lambda)$
 $(c^*, r^*) \leftarrow \mathcal{A}^E(\text{id})$
 $r \leftarrow E(c^*)$

Return 1 iff $\text{dis}(r, r^*) \leq d_{noise}$ and c^* is fresh.

For the sake of simplicity we use this game based definition of unpredictability. A formulation with respect to entropy contained in the PUF responses is given in [4]. A comprehensive and more in-depth game-based formulation of PUF properties is found in [2].

2.2 Definition of Logically Reconfigurable PUFs

In practice, many PUF instances have only a restricted challenge and response space, such that after a certain number of queries they cannot be used anymore. The basic idea of Logically Reconfigurable PUF (LR-PUFs) is to extend the PUF by a control logic that allows to change the challenge and response behavior of the system. Our definition is similar to the one of [8].

Definition 3 (Logically Reconfigurable PUFs). *Let $\mathcal{P} = (S, E)$ be a family of (ρ, d_{noise}) -PUFs. A logically reconfigurable PUF (LR-PUF) with black-box access to \mathcal{P} is a tuple of efficient algorithms $\mathcal{L} = (\text{Setup}^{S,E}, \text{Query}^E, \text{Rcnf}^E)$, which satisfies the following properties*

Setup. *The Setup algorithm takes as input the security parameter 1^λ . It outputs an index $\text{id} \in \mathcal{I}$, determining the underlying PUF from \mathcal{P} and an initial state $\text{st} \in \{0, 1\}^{\ell(\lambda)}$ of the LR-PUF. We require that $\ell(\lambda) \geq \lambda$.*

Query mechanism. *The Query_{st} algorithm takes as input a challenge $c \in \{0, 1\}^\lambda$ and outputs a response $r \in \rho(\lambda)$.*

Reconfiguration. *The Rcnf_{st} algorithm updates the state of the LR-PUF to a new state $\text{st}' \in \{0, 1\}^{\ell(\lambda)}$ which is (possibly probabilistically) computed from the old state st . The new state is also output.*

The three algorithms may interact with the underlying PUF family via the oracles E and S. We will often omit giving the oracle access explicitly. Additionally, we assume that the noise of the LR-PUF responses is bounded in the same way as the noise of the underlying PUF's responses.

Remark 1. The setup algorithm of almost all constructions in this paper is the same and consists of the following steps. $\text{Setup}(1^\lambda)$ generates the underlying PUF $\text{id} \leftarrow_{\S} \mathcal{S}(1^\lambda)$, chooses a string $\text{st} \leftarrow_{\S} \{0, 1\}^{\ell(\lambda)}$ uniformly at random, and outputs (id, st) . In the following, unless stated otherwise, all constructions will use this standard setup algorithm.

Unpredictability of LR-PUFs. Ideally, an LR-PUF in one specific state should be as unpredictable as its underlying physical PUF, so the internal state of the LR-PUF can be seen as a mapping from LR-PUF queries to PUF queries that is ideally a permutation. Reconfiguration then constitutes a “shuffling” of this mapping, such that a completely new permutation is reached. To formalize this, the authors of [8] propose two complimentary notions of unpredictability:

Forward-unpredictability: The reconfiguration changes the mapping in such a way, that knowledge about the previous state does not enable an adversary to predict the challenge-response behavior for the reconfigured LR-PUF.

Backward-unpredictability: The reconfiguration reveals no additional information about the old internal state, i.e., after reconfiguration an adversary should not be able to predict the challenge-response behavior for the old state.

We provide a formal characterization of these properties as derivatives of our plain unpredictability notion (see Definition 4).

3 New Notions of Unpredictability

In this section we extend the original unpredictability notion by considering strengthened adversaries. We show how the new notions relate to each other and in which scenarios their consideration might be beneficial.

3.1 Multiple Reconfigurations

In [8], the unpredictability experiments revolve around a single reconfiguration process. However, an adversary might witness several reconfigurations and thereby deduce some information about the influence of the state on the LR-PUFs behavior. This motivates our first unpredictability definition, which is an extension of the backward/forward-unpredictability properties to multiple reconfigurations of the LR-PUF. To this end, we provide the adversary access to a reconfiguration oracle, which invokes the reconfiguration algorithm.

Let Rcnf denote the reconfiguration oracle for an LR-PUF \mathcal{L} with current state st . The oracle Rcnf accepts two kinds of inputs: \perp , upon which Rcnf_{st} is invoked, and st' upon which the internal state of \mathcal{L} is set to st' . The latter input functionality allows the adversary to program the state and is only available to state-setting adversaries (see Appendix C.2). Let \mathcal{S} denote the list of states the adversary obtains over the course of an experiment, be it through Setup or the oracle Rcnf . Further, let Query denote the query oracle, which takes as input a state st and a challenge c and returns $\text{Query}_{\text{st}}(c)$ to the adversary. The adversary can only invoke the query oracle with states stored in \mathcal{S} .

Definition 4 (Plain-Unpredictability). An LR-PUF $\mathcal{L} = (\text{Setup}, \text{Query}_{\text{st}}, \text{Rcnf}_{\text{st}})$ is unpredictable if for any PPT adversary \mathcal{A} the probability that the experiment $\text{PLAIN}_{\mathcal{A}}^{\mathcal{L}}(\lambda)$ evaluates to 1 is negligible (in the security parameter λ), where the game is defined in Fig. 3.

Remark 2. We can obtain the backward- and forward-unpredictability notions described in [8] by considering restricted adversaries that invoke $\text{Rcnf}(\perp)$ only once, setting and obtaining the new state st' , in the following only query $\text{Query}(\text{st}', \cdot)$.

Corollary 1. Let \mathcal{L} be a Plain-unpredictable LR-PUF, then \mathcal{L} is also backward- and forward-unpredictable.

Separation of Plain- and Backward/Forward-Unpredictability. In what follows we show that Plain-unpredictability is strictly stronger than the previous notions. Let $\text{Backward}_{\mathcal{A}}^{\mathcal{L}}(\lambda)$ (resp. $\text{Forward}_{\mathcal{A}}^{\mathcal{L}}(\lambda)$) denote $\Pr[\text{PLAIN}_{\mathcal{A}}^{\mathcal{L}}(\lambda) = 1]$ where \mathcal{A} is a backward-unpredictability adversary (resp. forward-unpredictability adversary) as described above. We separate the security notions with the following two propositions.

Proposition 1. If collision-resistant hash functions relative to a PUF exist (cf. Appendix A), then there exist backward-unpredictable LR-PUFs, which are not Plain-unpredictable.

The basic idea of our counterexample is to let the adversary learn a prediction by calling the reconfiguration oracle. This prediction, however, only helps him in combination with the evaluation oracle, that outputs the evaluation on a point

Experiment $\text{PLAIN}_{\mathcal{A}}^{\mathcal{L}}(\lambda)$
 $(\text{id}, \text{st}) \leftarrow \text{Setup}(1^\lambda)$
 $(\text{st}^*, c^*, r^*) \leftarrow \mathcal{A}^{\text{Query}, \text{Rcnf}(\perp)}(1^\lambda, \text{st})$
 $r \leftarrow \text{Query}_{\text{st}^*}(c^*)$
 Output 1 iff $\text{dis}(r, r^*) \leq d_{\text{noise}}$ and c^*
 was not queried to $\text{Query}(\text{st}^*, \cdot)$
 and $\text{st}^* \in \mathcal{S}$.

Fig. 3. Security of plain unpredictability.

Experiment $\text{ST-ACCESS}_{\mathcal{A}}^{\mathcal{L}}(\lambda)$
 $(\text{id}, \text{st}) \leftarrow \text{Setup}(1^\lambda)$
 $(\text{st}^*, c^*, r^*) \leftarrow \mathcal{A}^{\text{Query}, \text{Rcnf}}(1^\lambda, \text{st})$
 $r \leftarrow \text{Query}_{\text{st}^*}(c^*)$
 Output 1 iff $\text{dis}(r, r^*) \leq d_{\text{noise}}$ and
 c^* was not queried to $\text{Query}(\text{st}^*, \cdot)$
 and $\text{st}^* \in \mathcal{S}$.

Fig. 4. Security of state-setting unpredictability.

Experiment $\text{PUF-ACCESS}_{\mathcal{A}}^{\mathcal{L}}(\lambda)$
 $(\text{id}, \text{st}) \leftarrow \text{Setup}(1^\lambda)$
 $(\text{st}^*, c^*, r^*) \leftarrow \mathcal{A}^{\text{Rcnf}(\perp), \text{E}}(1^\lambda, \text{st})$
 Set LR-PUF state to st^* using $\text{Rcnf}(\text{st}^*)$
 $\text{st}' \leftarrow \text{Rcnf}_{\text{st}^*}$
 $r \leftarrow \text{Query}_{\text{st}'}(c^*)$
 Output 1 iff $\text{dis}(r, r^*) \leq d_{\text{noise}}$.

Fig. 5. Security of direct access unpredictability.

Experiment $\text{FULL-ACCESS}_{\mathcal{A}}^{\mathcal{L}}(\lambda)$
 $(\text{id}, \text{st}) \leftarrow \text{Setup}^{\text{S}, \text{E}}(1^\lambda)$
 $(\text{st}^*, c^*, r^*) \leftarrow \mathcal{A}^{\text{Rcnf}, \text{E}}(1^\lambda, \text{st})$
 Set LR-PUF state to st^* using $\text{Rcnf}(\text{st}^*)$
 $\text{st}' \leftarrow \text{Rcnf}_{\text{st}^*}$
 $r \leftarrow \text{Query}_{\text{st}'}(c^*)$
 Output 1 iff $\text{dis}(r, r^*) \leq d_{\text{noise}}$.

Fig. 6. Security of full access unpredictability.

different from the challenge, if the query contains a specific prediction. More precisely, we store a pair (u, v) in the state. Then we modify the query algorithm, whose input is a challenge c , such that it evaluates the PUF on $(1^\lambda \oplus c || \text{st}')$ if $c = E(u)$. Clearly, in our construction the attacker can never invoke the query oracle on u and thus, cannot exploit this part directly. However, whenever the attacker queries the reconfiguration oracle, it obtains this answer through the state.

A more detailed proof appears in the full version of this paper.

Proposition 2. *If collision-resistant hash functions relative to a PUF (cf. Appendix A) exist, then there exist forward-unpredictable LR-PUFs, which are not Plain-unpredictable.*

The proof is analogous to the one of Proposition 1, as the same construction describe there is also forward-unpredictable and is thus omitted.

3.2 State-Setting Adversaries

The authors of [8] assume that the state is stored in a tamper-evident manner and therefore an attacker cannot set the state of the LR-PUF to arbitrary values. We believe that there are many plausible scenarios where tamper-evident storage of the state is too expensive and where the adversary might be able to change the state, even though the internal physical PUF is tamper-evident. Therefore, we propose the following unpredictability notion, in which an adversary can set the internal state of the LR-PUF. As mentioned above, this is formalized via the type of inputs to the reconfiguration oracle Rcnf which are arbitrary states that are to be set as the new state of the LR-PUF.

Definition 5 (St-Access-Unpredictability). *An LR-PUF $\mathcal{L} = (\text{Setup}, \text{Query}_{\text{st}}, \text{Rcnf}_{\text{st}})$ is unpredictable for a state-setting adversary if for any PPT adversary \mathcal{A} the probability that the experiment $\text{ST-ACCESS}_{\mathcal{A}}^{\mathcal{L}}(\lambda)$ evaluates to 1 is negligible (in the security parameter λ), where the game is defined in Fig. 4.*

The state-setting adversary can be thought of as bypassing the reconfiguration algorithm, thus security against state-setting adversaries should be considered a property of the Query mechanism.

Remark 3. It is easy to see that an LR-PUF construction satisfying this notion of unpredictability must also be **Plain-unpredictable** (Definition 4). Any adversary against **Plain-unpredictability** is also a valid adversary in the **ST-ACCESS-unpredictability** experiment, which simply does not invoke the reconfiguration oracle on an input other than \perp .

Corollary 2. *Let $\mathcal{L} = (\text{Setup}, \text{Query}_{\text{st}}, \text{Rcnf}_{\text{st}})$ be **St-Access-unpredictable**. Then \mathcal{L} is also **Plain-unpredictable**.*

The inverse relationship does, however, not hold.

Proposition 3. *There exist **Plain-unpredictable LR-PUF** constructions, which are not **St-Access-unpredictable**.*

To show the separation, we consider a construction which has a “vulnerable” state, i.e., a state which does not support a secure reconfiguration. A state-setting adversary can then prepare the LR-PUF to have that state and get an advantage through the defective reconfiguration algorithm. An adversary without state-setting capabilities, however, would have to wait for that state to occur in a chain of honest reconfigurations to get any advantage, as long as the reconfiguration is working correctly for any other state. For more details, please refer to the full version of this paper.

3.3 Direct Access Adversaries

Another assumption made in [8] is that the attacker cannot bypass the Query mechanism and thus, does not have direct access to the embedded PUF. In the real world, however, it might be that case that the attacker finds a way to stimulate the physical PUF directly, circumventing the control logic of the LR-PUF. In what follows, we remove this assumption by giving the adversary direct access to the embedded PUF as well.

Definition 6 (PUF-Access-Unpredictability). *An LR-PUF $\mathcal{L} = (\text{Setup}, \text{Query}_{\text{st}}, \text{Rcnf}_{\text{st}})$ is unpredictable for an adversary with direct PUF access if for any PPT adversary \mathcal{A} the probability that the experiment $\text{PUF-ACCESS}_{\mathcal{A}}^{\mathcal{L}}(\lambda)$ evaluates to 1 is negligible (in the security parameter λ), where the game is defined in Fig. 5.*

Because an LR-PUF construction might rely solely upon the PUF itself to perform reconfiguration and querying, an adversary that has access to the PUF may be able, given the current state of the PUF, to compute challenge-response pairs for all the following states the LR-PUF will have.

Proposition 4. *If collision-resistant hash functions relative to a PUF exist, then there exists a **Plain-unpredictable LR-PUF** construction, which is not **PUF-Access-unpredictable**.*

The proof relies on the fact that an adversary can in some construction simulate the Rcnf and Query oracles himself. The full proof can be found in the full version of this paper.

Perhaps surprisingly there is an obstacle when trying to compare the power of state-setting adversaries to PUF-access adversaries. As described above, a PUF-access adversary might be able to completely precompute the behavior of an LR-PUF given the current state. Thus the definition of **PUF-Access-unpredictability** demands the adversary predict a challenge response pair not for the state, which it finally outputs, but for the state which results from the reconfiguration based on that state. This excludes bypassing the Rcnf oracle and enables the definition to capture the unpredictability gain provided by the Rcnf algorithm.

Proposition 5. *Unpredictability against state-setting adversaries is not comparable to unpredictability against PUF-access adversaries, i.e.,*

- (i) *There exists an LR-PUF which is **PUF-Access-unpredictable** but not **St-Access-unpredictable**.*
- (ii) *If collision-resistant hash functions w.r.t. PUFs exist, there exist LR-PUFs, which are **St-Access-unpredictable** but not **PUF-Access-unpredictable**.*

The proof of this proposition appears in the full version of this paper.

3.4 Full Access Adversaries

A combination of the previous scenarios provides the PUF access adversary with the possibility to set the internal state. This is intuitively the strongest notion, as it provides the adversary with essentially complete control over the LR-PUF during the query phase of the experiment.

Definition 7 (Full-Access-Unpredictability). *An LR-PUF $\mathcal{L} = (\text{Setup}, \text{Query}_{\text{st}}, \text{Rcnf}_{\text{st}})$ is unpredictable for a state-setting adversary with PUF access if for any PPT adversary \mathcal{A} the probability that the experiment $\text{FULL-ACCESS}_{\mathcal{A}}^{\mathcal{L}}(\lambda)$ evaluates to 1 is negligible (in the security parameter λ), where the game is defined in Fig. 6.*

As **Full-Access-unpredictability** is an immediate extension of **PUF-Access-unpredictability**, it is easy to see that any **Full-Access-unpredictable** \mathcal{L} is also **PUF-Access-unpredictable**. However, the **PUF-Access-unpredictability** adversary is strictly weaker than the state-setting **Full-Access-unpredictability** adversary.

Proposition 6. *There are LR-PUF constructions which are **PUF-Access-unpredictable**, but not **Full-Access-unpredictable**.*

For the full proof, please refer to the full version of this paper.

Since **Full-Access-unpredictability** implies **PUF-Access-unpredictability** and we because have seen that **St-Access-** and **PUF-Access-unpredictability** do not imply each other (see Proposition 5), **Full-Access-unpredictability** can also not follow from **St-Access-unpredictability**.

Corollary 3. *There exists a **St-Access-unpredictable** LR-PUF construction, which is not secure w.r.t. **Full-Access-unpredictability**.*

4 Construction

In this section we present our construction that fulfills the **Full-Access** notion of unpredictability we defined in Sect. 3. Our scheme can be seen as a randomized version of the speed-optimized construction from [8] with the difference that the reconfiguration algorithm chooses a fresh state uniformly at random (instead of computing it as the hash of the old state). Afterwards, we show that the reconfiguration algorithm must be randomized in order to achieve our strongest notion of unpredictability.

Query _{st} (<i>c</i>) $w \leftarrow \text{Hash}(\text{st} c)$ $y \leftarrow E(w)$ Return <i>y</i>	Rcnf _{st} $\text{st} \leftarrow_{\mathcal{S}} \{0, 1\}^{\ell(\lambda)}$ Return <i>st</i>
--	---

Fig. 7. The full LR-PUF construction.

Theorem 1. *The full construction (Fig. 7) is **Full-Access-unpredictable**.*

As the reconfiguration algorithm chooses a new state uniformly at random, the probability that the adversary correctly predicts the new state is negligible. The probability that the output prediction made by the adversary is valid for a different state as well can be bounded by the probability of predicting the outputs of the underlying physical PUF, which was assumed to be negligible. The full proof of this theorem appears in the full version of this paper.

Proposition 7. *The full construction is **St-Access-unpredictable**.*

As the construction's **Query** algorithm is the same as the **speed**-construction's (see Fig. 8), and the **Rcnf**-algorithm cannot be used in any advantageous way by an adversary, the construction is **St-Access-unpredictable** as long as the **speed**-construction is **St-Access-unpredictable**. This is shown in Proposition 9. The proof of this theorem appears in the full version of this paper.

5 Conclusion

In this paper, we have reconsidered the concept of Logically Reconfigurable PUFs, an extension of the PUF primitive with applications in embedded devices for access control or object tracking. We have given a formal definition of LR-PUFs and presented several new notions of unpredictability, which help to classify constructions according to the scenarios they could be employed in. An evaluation of two previously given construction has shown these constructions to withstand stronger adversaries than initially shown. Finally, we have given a new construction that can handle the strongest adversaries defined in this work and we have seen that these notions create an interesting separation between such constructions that rely on deterministic reconfiguration algorithms and such that randomize reconfiguration.

Acknowledgements. Dominique Schröder was supported by the German Federal Ministry of Education and Research (BMBF) through funding for the Center for IT-Security, Privacy and Accountability (CISPA www.cispa-security.org) and also by an Intel Early Career Faculty Honor Program Award. Finally, we thank the reviewers for their valuable comments.

A PUFs and Collision-Resistant Hash Functions

As some of the LR-PUF constructions we will discuss use collision resistant hash functions, we will first have to define this primitive.

Definition 8 (Hash Function). *Let $\ell : \mathbb{N} \rightarrow \mathbb{N}$ be a polynomial. A pair of PPT algorithms $(\text{Gen}, \text{Hash})$ is called a hash function if:*

Gen. *Takes as input a security parameter 1^λ and returns an index id from some index set \mathcal{I}_λ .*

Hash. *Takes as input an index id and a bit string $x \in \{0, 1\}^*$. It returns an output string $\text{Hash}(\text{id}, x) \in \{0, 1\}^{\ell(\lambda)}$. We set $\text{Hash}(x) := \text{Hash}(\text{id}, x)$.*

What follows is the usual definition of collision resistance against a PPT adversary [7], modified to account for possible advantages an adversary might have through PUF Access. For a discussion of the possible complexity theoretic implications of PUF access, please refer to [4, 12].

Definition 9 (Collision Resistance w.r.t. PUFs). *Let $\mathcal{P} = (\text{S}, \text{E})$ be a PUF family. A hash function is called collision-resistant with respect to \mathcal{P} , if for all PPT algorithms \mathcal{A} that have black-box access to \mathcal{P} the probability that the experiment $\text{COLL}_{\mathcal{A}}^{\mathcal{P}}$ evaluates to 1 is negligible, where*

$$\begin{aligned} & \textbf{Experiment } \text{COLL}_{\mathcal{A}}^{\mathcal{P}}(\lambda) \\ & \quad \text{id} \leftarrow \text{Gen}(1^\lambda) \\ & \quad (x, x') \leftarrow \mathcal{A}^{\text{S}, \text{E}}(\text{id}) \\ & \quad \text{Return } 1 \text{ iff } \text{Hash}(x) = \text{Hash}(x') \text{ and } x \neq x'. \end{aligned}$$

A.1 PUFs and Asymptotic Security

From a practitioners perspective, the asymptotic formulation of unpredictability presented in this paper can seem problematic, as it does not quantify exactly how much security a given construction provides. In [14] there is also an argument made against asymptotic security claims about PUFs which suggests that such claims are not meaningful. The core of the argument is that PUFs are finite functions and an adversary could just have a hard-coded table of challenge response pairs for a given PUF. We believe this criticism is not applicable in our case, as we define PUFs as families of functions from which one is sampled in the unpredictability experiment. As the probability is also taken over the randomness of this sampling process, it is unlikely that the adversary will have the lookup table for *this specific PUF* hard-coded.

B Deterministic Reconfiguration Algorithms

As seen before, the PUF-access variants of unpredictability exclude LR-PUF constructions where access to the underlying PUF makes the Query and Rcnf algorithms completely predictable to the adversary. We will now show that an LR-PUF construction, which is, in this sense, deterministic cannot achieve PUF-access unpredictability.

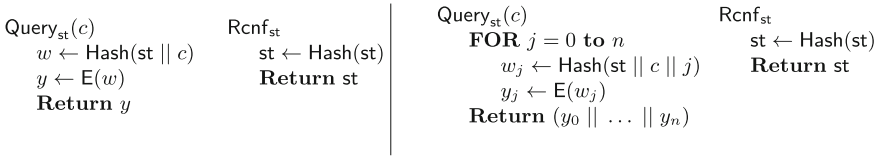


Fig. 8. The speed-optimized construction `speed` and the area-optimized construction `area` from [8].

Proposition 8. *A LR-PUF construction $\mathcal{L} = (\text{Setup}, \text{Query}_{\text{st}}, \text{Rcnf}_{\text{st}})$, where Rcnf_{st} is deterministic cannot achieve **PUF-Access-unpredictability**.*

The proof appears in the full version of this paper.

As **PUF-Access-unpredictability** is implied by **Full-Access-unpredictability**, the following corollary follows immediately.

Corollary 4. *An LR-PUF construction $\mathcal{L} = (\text{Setup}, \text{Query}_{\text{st}}, \text{Rcnf}_{\text{st}})$, where Rcnf_{st} is deterministic cannot achieve **Full-Access-unpredictability**.*

Of course, these claims exclude LR-PUF constructions which involve more than one PUF, as the **PUF-Access-** and **Full-Access-unpredictability** experiments only provide access to a single underlying PUF. It is, however, straightforward to extend the definition to multiple PUFs. This can also be intuitively motivated, as a multi-PUF construction, which does not secure access to one of the employed PUFs is unlikely to secure access to the other PUFs.

Remark 4. As generation of proper randomness on an highly embedded system such as a public transport access token seems impractical, this result establishes that the **Full-Access-unpredictability** notion might not be achievable for all application scenarios. However, in scenarios, where reconfigurations occur infrequently the negative effect of randomness that comes from a weak source is likely to be tolerable. Additionally, if the degree of sophistication of the device is high enough, there could already be a circuit in the device implementing a hash function such as SHA-2, which may be used to extract some randomness in a heuristic fashion.

C Revisiting Earlier Constructions

Based on the newly proposed unpredictability notions it is worthwhile to revisit the original LR-PUF constructions given in [8]. We show that they provide unpredictability in more adverse settings than originally demonstrated.

C.1 Speed-Optimized-Construction

Let us first consider the *speed-optimized* implementation of LR-PUFs given in [8]. The basic idea of the construction is to use a collision-resistant hash

function to compound the state and the LR-PUF challenge into one PUF stimulus. The formal description of the algorithms is shown in the left part of Fig. 8. As the reconfiguration algorithm is deterministic, this construction cannot achieve **PUF-Access-** or **Full-Access-**unpredictability (see Sect. 4), however, we can show that it is **St-Access-**unpredictable, which is an improvement on the unpredictability result given in [8].

Proposition 9. *The speed construction is **St-Access-**unpredictable.*

The proof can be found in the full version of this paper.

C.2 Area-Optimized-Construction

Beside the speed-optimized construction, [8] also propose an area-optimized construction, in which a small-range PUF is stimulated repeatedly on different sub-challenges derived from the original challenge. The sub-challenge responses are then assembled to one larger response to the original challenge. The formal description is given in the right part of Fig. 8. We will now see that the construction *can* be **St-Access-**unpredictable, but the degree of unpredictability depends on the choice of underlying PUF and the iteration count n . Note, that a noise bound d_{noise} on the full response means that the underlying PUF should not produce responses that are noisier than $\frac{d_{noise}}{n}$.

Proposition 10. *If there exist collision-resistant hash functions with respect to PUFs, and the underlying PUF is unpredictable, then the area-construction is **St-Access-**secure.*

For the proof of this statement, please refer to the full version of this paper.

References

1. Akdemir, K.D., Wang, Z., Karpovsky, M., Sunar, B.: Design of cryptographic devices resilient to fault injection attacks using nonlinear robust codes. In: Joye, M., Tunstall, M. (eds.) *Fault Analysis in Cryptography. Information Security and Cryptography*, pp. 171–199. Springer, Berlin Heidelberg (2012)
2. Armknecht, F., Maes, R., Sadeghi, A.-R., Standaert, F.-X., Wachsmann, C.: A formalization of the security features of physical functions. In: 2011 IEEE Symposium on Security and Privacy, pp. 397–412, Berkeley, California, USA. IEEE Computer Society Press, 22–25 May 2011
3. Armknecht, F., Maes, R., Sadeghi, A.-R., 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)
4. 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)
5. Eichhorn, I., Koeberl, P., van der Leest, V.: Logically reconfigurable pufs: memory-based secure key storage. In: *Proceedings of the Sixth ACM Workshop on Scalable Trusted Computing*, pp. 59–64. ACM (2011)

6. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Controlled physical random functions. In: Proceedings of the 18th Annual Computer Security Conference (2002)
7. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. CRC Press, Boca Raton (2008)
8. Katzenbeisser, S., Koçabas, Ü., van der Leest, V., Sadeghi, A.-R., Schrijen, G.-J., Schröder, H., Wachsmann, C.: Recyclable PUFs: logically reconfigurable PUFs. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 374–389. Springer, Heidelberg (2011)
9. Škorić, B., Tuyls, P., Oprey, W.: Robust key extraction from physical unclonable functions. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 407–422. Springer, Heidelberg (2005)
10. Kursawe, K., Sadeghi, A., Schellekens, D., Skoric, B., Tuyls, P.: Reconfigurable physical unclonable functions - enabling technology for tamper-resistant storage. In: IEEE International Workshop on Hardware-Oriented Security and Trust, HOST 2009, pp. 22–29 (2009)
11. Maes, R., Verbauwhede, I.: Physically unclonable functions: a study on the state of the art and future research directions. In: Sadeghi, A.-R., Naccache, D. (eds.) Towards Hardware-Intrinsic Security. Information Security and Cryptography, pp. 3–37. Springer, Berlin Heidelberg (2010)
12. Ostrovsky, R., Scafuro, A., Visconti, I., Wadia, A.: Universally composable secure computation with (malicious) physically unclonable functions. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 702–718. Springer, Heidelberg (2013)
13. Pappu, R.S.: Physical one-way functions. PhD thesis (2001)
14. Rührmair, U., Sölter, J., Sehne, F.: On the foundations of physical unclonable functions. Cryptology ePrint Archive, Report 2009/277 (2009). <http://eprint.iacr.org/>
15. Sadeghi, A.-R., Visconti, I., Wachsmann, C.: PUF-enhanced RFID security and privacy. In: Secure Component and System Identification (SECSI), Cologne, Germany, April 2010
16. Schulz, S., Sadeghi, A.-R., Wachsmann, C.: Short paper: lightweight remote attestation using physical functions. In: Proceedings of the Fourth ACM Conference on Wireless Network Security, WiSec 2011, pp. 109–114. ACM, New York, NY, USA (2011)
17. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: Proceedings of the 44th Annual Design Automation Conference, DAC 2007, pp. 9–14. ACM, New York, NY, USA (2007)
18. Tuyls, P., Batina, L.: RFID-tags for anti-counterfeiting. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 115–131. Springer, Heidelberg (2006)