Strong PUFs: Models, Constructions and Security Proofs

Ulrich Rührmair and Heike Busch and Stefan Katzenbeisser

Abstract Classical authentication and identification protocols are commonly based on the possession of a secret key. It is assumed that this key does not fall in the hands of an adversary. However, this assumption may be violated if the adversary has physical access to the device that performs authentication. He may read out the memory of the device, including all secret information. Even if dedicated hardware security measures are in use, they may be circumvented by specialized attacks such as side-channel analysis or invasive methods. Alternatively, software attacks such as viruses can reveal key material. Physically Unclonable Functions (PUFs) were proposed to mitigate this risk. They avoid direct storage of digital binary keys in hardware systems, providing inexpensive and lightweight security solutions. In this chapter, we investigate the powerful concept of a Strong PUF more closely. We first give an overview of Strong PUF implementations that can be used as basic building blocks in authentication and identification protocols. Subsequently we turn to the formal foundations of Strong PUFs. We analyze existing definitions, and introduce new semi-formal and formal adversarial models. We then perform a security proof for a Strong PUF-based identification scheme in one of our models.

1 Introduction

Electronic devices have pervaded our everyday life to a previously unseen extent, and will likely continue to do so in the future. But their ubiquity also makes them

Ulrich Rührmair
Technische Universität München, e-mail: ruehrmai@in.tum.de

Heike Busch
Technische Universität Darmstadt, e-mail: busch@seceng.informatik.tu-darmstadt.de

Stefan Katzenbeisser
Technische Universität Darmstadt, e-mail: katzenbeisser@seceng.informatik.tu-darmstadt.de

a potential target for adversaries, and brings about privacy and information security issues.

The tools that classical cryptography offers in order to fight these issues all rest on the concept of a secret binary key. They assume that devices can contain a piece of information that is, and remains, unknown to the adversary. Unfortunately, this assumption can be difficult to uphold in practice: Physical attacks such as invasive, semi-invasive, or side-channel attacks, as well as software attacks like API-attacks and viruses, can lead to key exposure. The fact that the employed devices should be inexpensive, mobile and cross-linked aggravates the problem.

The described situation was one of several motivations that inspired researchers to develop the concept of a *Physical Unclonable Function* (PUF). A PUF is a physical system S that can be challenged with so-called stimuli or challenges C_i, and which reacts with corresponding responses R_{C_i}. The responses shall depend on manufacturing variations or structural disorder in S that is beyond the control of the original manufacturer, and which cannot be cloned or reproduced exactly. The tuples (C_i, R_{C_i}) are often called *challenge-response pairs* (CRPs) of the PUF.

Two important subtypes of PUFs are so-called *Strong PUFs* and *Key Obfuscating PUFs* [20]; the latter have also been called Weak PUFs in [8] or Physically Obfuscated Keys (POKs) in [6]. Strong PUFs must possess a very large number of possible challenges. A complete determination/measurement of all challenge-response pairs within a limited time frame (such as several days or even weeks) must be impossible. Furthermore, it must be difficult for an adversary to numerically predict or guess the response R_C of a Strong PUF to a randomly selected challenge C. This should hold even if many other challenge-response pairs are known to him. Thus, a Strong PUF's challenge-response behavior must be complex, and difficult to imitate and "learn". A well-known example of a Strong PUF is the Optical PUF of [15, 16], which also historically is the first PUF that has been suggested.

Typical applications of Strong PUFs are key establishment and identification protocols [16]. The latter usually work in the following manner: A central authority (CA) holds a secret list of many CRPs of a PUF S. The PUF is assumed to be embedded in a hardware system or contained on a security token. In order to identify the PUF, the CA sends k randomly chosen challenges $C_1, \ldots, C_k$ from the CRP list. If the hardware/token can return the correct, corresponding PUF-responses $R_{C_1}, \ldots, R_{C_k}$, then the identification is successful. Note that such an approach avoids the storage of secret binary keys in the PUF-embedding hardware. It also avoids the use of standard symmetric or asymmetric cryptosystems, whose security depends on a small set of well-known, but unproven assumptions. It also obviates the potentially costly implementation of standard cryptosystems in mobile devices.

Key Obfuscating PUFs, on the other hand, have few challenges—in the extreme case just one, fixed challenge. Their response(s) are used to derive a classical binary secret key, which is subsequently processed by the embedding system in a standard fashion, i.e. as a secret input for classical cryptosystems. This makes Key Obfuscating PUFs similar to a non-volatile key storage. Their advantage is that they may be harder to read out invasively than common non-volatile memory such as EEPROM. Since they depend on inherent manufacturing variations, they can individu-

alize hardware without costly, dedicated individualization steps in the production. Typical examples of Key Obfuscating PUFs are the SRAM PUF [8], Butterfly PUF [10] and Coating PUF [21].

Since Key Obfuscating PUFs are nothing else than a special form of secret key storage, they can be used for essentially all cryptographic schemes and applications. Please note, however, that this also makes them susceptible to side channel attacks like power consumption or emanation analysis, in just the same manner as classical schemes. Protocols based on Key Obfuscating PUFs usually show the same dependency on computational assumptions as standard cryptoschemes built on secret binary keys. Furthermore, since zero errors in the derivation of the secret key from the PUF are tolerable, error correction plays a much more critical role for Key Obfuscating PUFs.

In this paper, we focus on Strong PUFs, and investigate their formal foundations and their application for identification purposes. We start by an overview of currently existing Strong PUF implementations in Section 2. Then, we analyze currently existing definitions of (Strong) PUFs in Section 3, and devise new adversarial models and definitions in Section 4. We introduce PUF-based identification schemes in Section 5. Subsequently, we perform a formal security proof for identification based on Strong PUFs in one of our models. We conclude the paper in Section 6.

2 Implementations of Strong Physical Unclonable Functions

We start by surveying the current candidates for Strong Physical Unclonable Functions. In 2001, Pappu [15] suggested an optical system as the historically first PUF. It consists of a laser beam, which is directed at a transparent scattering token comprising of many randomly distributed scatterers. The laser light is scattered multiple times in the token, and interferes constructively and destructively with itself. This leads to an interference pattern of bright and dark spots on a subsequently placed CCD. This pattern sensitively depends on the location of the scatterers in the token, but also on the angle and point of incidence of the laser light (and on other parameters of the set-up).

The angle and point of incidence of the laser beam are usually regarded as the challenge of this PUF, while the interference pattern (or a suitably chosen image transformation of it) is interpreted as its response. This optical Strong PUF offers high internal complexity and security. On the downside, it cannot be integrated easily into an electronic microsystem, and requires an external, precise read-out apparatus.

Relatively soon afterwards, integrated, electrical candidates for Strong PUFs have been suggested. One important example is the so-called Arbiter PUF [5, 13], which exploits the natural variations in the runtime delays of integrated circuits. The Arbiter PUF consists of a sequence of k stages (e.g. multiplexers), which are conditioned by a corresponding sequence of k external bits $(b_1, \ldots, b_k)$. An incoming electrical signal is split into two signals, which race against each other in parallel

through the sequence of stages. Their exact paths are thereby determined by the values b_i. At the end of the structure, an "arbiter element" (consisting of a latch) determines whether the top or bottom path arrived first, and correspondingly outputs a zero or a one. The Arbiter PUF thus maps a k-bit input challenge $C_i = (b_1, \ldots, b_k)$ to a 1-bit response R_{C_i}.

However, it has been noted early by its inventors that the Arbiter PUF can be attacked successfully by standard machine learning (ML) methods, such as Support Vector Machines or Perceptrons [5, 12]. An attacker collects a number of CRPs and uses them to train the ML algorithm. If trained successfully, the algorithm will subsequently be able to predict the correct responses to other challenges with high probability.

In order to improve their resilience to ML attacks, several variants of the basic Arbiter PUF have been proposed. Examples include XOR Arbiter PUFs [4], Feed-Forward Arbiter PUFs [5, 11], and Lightweight Secure PUFs [14]. All of them are based on runtime delays, but employ the basic Arbiter PUF as a building block in more complex architectures. Nevertheless, it has been shown recently that even these improved variants can be attacked by more sophisticated ML techniques [19, 20], at least for instances of medium lengths. The critical question in the long term will be whether circuit implementations of Arbiter PUF variants can be made stable in size regimes that are beyond the reach of improved ML techniques.

Another potential Strong PUF candidate that must be considered is the Power Grid PUF of [9]. It exploits the resistance variations in the power grid of integrated circuits. A Power Grid PUF can in principle be used both as Key Obfuscating PUF and as Strong PUF. Due to its simple linear model, however, the Power Grid PUF is presumably susceptible to ML attacks just like other linear PUF structures. This makes it more useful as Key Obfuscating PUF, as already noted in [9].

Two approaches that follow new routes to machine learning resilient Strong PUFs have been suggested just recently. In [2, 3], analog circuits, in particular so-called Cellular Nonlinear Networks (CNNs), have been introduced as Strong PUFs. CNNs are two-dimensional, cellular, analog computing arrays, in which every cell is coupled in an analog fashion to its direct neighbors. Commercially available, programmable CNNs contain millions of transistors, while operating in a stable fashion. CNN-PUFs promise to allow stable PUFs with a very large number of interacting components, whose output strongly depends on a very large number of random components. Furthermore, their internal models are driven by complex differential equations, which complicates re-modeling and machine learning attacks.

A second recent approach to machine learning resistant Strong PUFs [17, 18] is to employ as many densely packed, independent random subunits as possible, which are read out individually and independently of each other at slow read-out rates. It was shown in [17] that large, monolithic, memory-like crossbar structures based on random diodes can practically implement this approach. They reach optimal information densities of up to 10^{10} bits per cm^2, and can be designed such that the slow read-out rate is not enforced by an artificially slow access module or the like, but by the inductive and resistive capacitances of the structure itself [17]. Faster read-out

leads to overloading and immediate destruction of the wiring, rendering the remaining structure unusable.

The resulting Crossbar PUFs are provably immune against machine learning and any other computational attacks. Their security merely depends on the access time of the adversary, and on the ratio of the already read-out bits vs. the number of overall bits stored in the structure. Modeling attacks subsequent to read-out are fruitless, since all components are independent of each other. Whether the limited read-out speed is a severe disadvantage depends on the intended application of this Strong PUF. [18] suggest the term SHIC PUFs (pronounce as *"chique PUFs"*) for this new category of Strong PUFs, where SHIC stands for Super High Information Content.

3 Physical Unclonable Functions: Towards a Formal Definition

In the following we take a closer look at formal definitions proposed for PUFs, which is a necessary prerequisite to being able to formally reason about the security of PUF-based protocols. Our discussion follows [20].

3.1 Physical One-Way Functions

We start our overview with the historically first definition, which is the definition of Physical One-Way Functions [15]. The following Notation 3.1 and Definition 3.2 are taken directly from [15].

Notation 3.1 (Notation for Physical One-Way Functions) *Let Σ be a physical system in an unknown state $X \in \{0,1\}^l$. X could also be some property of the physical system. l is a polynomial function of some physical resource such as volume, energy, space, matter, et cetera.*

Let $z \in \{0,1\}^k$ be a specific state of a physical probe P such that k is a polynomial function of some physical resource. Henceforth, a probe P in state z will be denoted by P_z.

Let $y = f(X, P_z) \in \{0,1\}^n$ be the output of the interaction between system Σ containing unknown state X and probe P_z.

Definition 3.2 (Physical One-Way Functions) $f : \{0,1\}^l \times \{0,1\}^k \to \{0,1\}^n$ *is a* PHYSICAL ONE-WAY FUNCTION *if*

- $\exists$ *a deterministic physical interaction between P and Σ which outputs y in $O(1)$, i.e. constant, time.*
- *Inverting f using either computational or physical means requires $\Omega(exp(l))$ queries to the system Σ.*
  This may be restated in the following way: The probability that any probabilistic polynomial time algorithm or physical procedure A' acting on $y = f(X, P_r)$,

where $y \in \{0,1\}^n$ is drawn from a uniform distribution, is able to output X or P_r is negligible. Mathematically,

$$Pr[A'(f(X,P_r)) \text{ outputs } X \text{ or } P_r] < \frac{1}{p(l)}$$

where $p(\,)$ is any positive polynomial. The probability is taken over several realizations of r.

We also stipulate that for any physical one-way function f

- *Simulating y, given X and P, requires either $O(poly(l))$ or $O(exp(l))$ in time/space resources depending on whether f is a* WEAK *or* STRONG *physical one-way function.*
- *Materially constructing a distinct physical system Σ' such that its unknown state $X' = X$ is hard.*

Definition 3.2 is reminiscent of the well-known definitions of mathematical one-way functions. It transfers the concepts known from that area (such as polynomial time and negligible probability) to the physical, finite context of PUFs. We would like to stress that the definition certainly owns the great merit of being the first formalization attempt in the field. It is associated with the highly respected, seminal work of [15] that established the whole field. But nevertheless, it touches upon a few noteworthy issues.

Firstly, let us address some formal aspects. Definition 3.2 employs the concept of polynomial resources and of negligible probability. However, these concepts cannot directly be applied to finite functions $f : \{0,1\}^l \to \{0,1\}^k$. In particular, no such function can meet the "hard to invert"-condition of Definition 3.2, since there is always an algorithm that contains f hard-coded as a lookup-table in its code. Since l and k are constant, this table has constant size; browsing the table for inverting f thus requires constant time as well.

These formal problems could be resolved by a suitable asymptotic treatment. Such a treatment might work along similar lines as collections of one-way functions [7], and could consider infinite families $(f_i)_{i \in I}$ of Physical One-Way Functions. Perhaps such a treatment was already attempted in Notation 3.1, when it is stated that l and k are a polynomial function of some physical resource. However, in Definition 3.2, l and k are treated as constants, and also the other relevant parameters of Definition 3.2, such as the polynomial runtime of A', are not functions of an external parameter.

Still, even if we assume that Definition 3.2 was meant to be asymptotic in some physical resource, and if l and k were intended to be functions of this resource, another interesting issue arises. The parameter n, which describes the length of $f(X,P_r)$, is a constant both in Definition 3.2 and Notation 3.1. At the same time, the runtime of A' is required to be polynomial in $|f(X,P_r)| = n = const$. This is not meaningful, since A' should be given longer computation time for a growing size of the considered PUF-instances. If Definition 3.2 was intended asymptotically, it might be better to formulate the non-invertability condition in the following manner:

$$Pr[A'(f(X,P_r),1^{k+l}) \text{ outputs } X \text{ or } P_r] < \frac{1}{p(l)}. \quad (1)$$

There are some interesting conceptual aspects of Definition 3.2 as well. For example, it can be observed that Definition 3.2 excludes functions with small ranges, e.g. ones with a binary output $\{0,1\}$. Such functions are not hard-to-invert in the sense of Definition 3.2. The reason is that for each of the two possible output values, *some* pre-image can always be found efficiently—simply by testing several randomly chosen challenges for their response until there is a match. Most electrical candidates for PUFs (e.g. variations of the Arbiter PUF or the Ring Oscillator PUF) have only a single bit or a fixed number of bits as output. They are hence excluded by Definition 3.2. Note that it cannot be regarded as a flaw of Definition 3.2 that it excludes electrical PUFs, as they were only introduced after the definition was written. Nevertheless, our observation points at two facts: (i) The concept of Physical One-Way Functions cannot serve as a comprehensive PUF-definition today. (ii) The non-invertibility condition of Definition 3.2 might not be the essential feature that makes PUF applications work. To our knowledge, the only PUF application where the non-invertability of f plays a role is the bit commitment protocol that was described in [15].

3.2 Physical Unclonable Functions

Another characterization of PUFs was given in [8]. It is not marked as a formal definition in the original text of [8], whence we term it a description here. It distinguishes between Strong PUFs and Weak PUFs, and is as follows.

Description 3.3 (Physical Unclonable Functions) *Physical Unclonable Functions consist of inherently unclonable physical systems. They inherit their unclonability from the fact that they consist of many random components that are present in the manufacturing process and can not be controlled. When a stimulus is applied to the system, it reacts with a response. Such a pair of a stimulus C and a response R is called a challenge-response pair (CRP). In particular, a PUF is considered as a function that maps challenges to responses.*

The following assumptions are made on the PUF:

1. *It is assumed that a response R_i (to a challenge C_i) gives only a negligible amount of information on another response R_j (to a different challenge C_j) with $i \neq j$.*
2. *Without having the corresponding PUF at hand, it is impossible to come up with the response R_i corresponding to a challenge C_i, except with negligible probability.*
3. *Finally, it is assumed that PUFs are tamper evident. This implies that when an attacker tries to investigate the PUF to obtain detailed information of its structure,*

the PUF is destroyed. In other words, the PUF's challenge – response behavior is changed substantially.

We distinguish between two different situations. First, we assume that there is a large number of challenge response pairs (C_i, R_i), $i = 1, \ldots, N$, *available for the PUF; i.e. a strong PUF has so many CRPs such that an attack (performed during a limited amount of time) based on exhaustively measuring the CRPs only has a negligible probability of success and, in particular,* $1/N \approx 2^{-k}$ *for large* $k \approx 100$. *We refer to this case as* **Strong PUFs**. *If the number of different CRPs N is rather small, we refer to it as a* **Weak PUF**. *Due to noise, PUFs are observed over a noisy measurement channel i.e. when a PUF is challenged with* C_i *a response* R_i' *which is a noisy version of* R_i *is obtained.*

Description 3.3 stipulates that Strong PUFs shall have an exponential number N of CRPs, with $1/N \approx 2^{-k}$ for some k with $k \approx 100$. In addition, item 1 of Description 3.3 demands that all CRPs of the PUF shall only reveal a negligible amount of information about each other. It is not fully clear how and under which conditions these two requirements can be met simultaneously. For example, it is argued in detail in [20] that the information content of any physical system is bounded polynomially in its size. If this is true, then the two above requirements mutually exclude each other. Again, this definition excludes PUFs whose output consists only of a single bit (such as the aforementioned Arbiter PUF and the Ring Oscillator PUF), as the probability to guess the PUF output correctly is at least $1/2$. This is better than negligible. Therefore, all these PUFs are excluded as Strong PUFs by Description 3.3.

The concept of Weak PUFs in the sense of Description 3.3 is logically consistent. But it is a relatively restrictive notion. From all currently known PUFs, only coating PUFs and SRAM-based PUFs are Weak PUFs. The reason is that (i) they only have very few possible challenges; and (ii) their responses to different challenges are fully independent of each other, since they read out single, non-interacting subunits of the PUF (isolated SRAM-cells in the case of SRAM PUFs and spatially isolated sensor arrays in the case of coating PUFs). Therefore the mutual information that different responses give about each other is essentially zero. For all other known PUFs (in particular the Arbiter PUF including all of its variants, Ring Oscillator PUFs, and Pappu's Optical PUFs), most responses to different challenges contain a non-negligible mutual amount of information about each other. This contradicts item 1 of Definition 3.3.

3.3 Physical Random Functions

Another PUF-definition, taken from [5], is as follows.

Definition 3.4 (Physical Random Functions) *A* PHYSICAL RANDOM FUNCTION (PUF) *is a function that maps challenges to responses, that is embodied by a physical device, and that verifies the following properties:*

1. *Easy to evaluate: The physical device is easily capable of evaluating the function in a short amount of time.*
2. *Hard to predict: From a polynomial number of plausible physical measurements (in particular, determination of chosen challenge-response pairs), an attacker who no longer has the device, and who can only use a polynomial amount of resources (time, matter, etc.) can only extract a negligible amount of information about the response to a randomly chosen challenge.*

The terms short and polynomial are relative to the size of the device. [1]

Definition 3.4 is very compact and intuitively appealing. It also stipulates some sort of asymptotic treatment, with the parameter being the size of the system. A few interesting conceptual aspects can be observed. The underlying security model allows an adversary to measure polynomially many challenge-response pairs (CRPs). This has the consequence that several PUFs cannot meet the definition, since they only possess polynomially many challenges at all. An adversary can create a full look-up table without breaking the polynomial CRP bound, and can use this table to imitate/predict the PUF. This applies, for example, to the Ring Oscillator PUF [4], which has only a quadratic number of challenges. It also holds for the Optical PUF of [15, 16]: Its number of CRPs is directly proportional to the dimensions of the scattering token, multiplied by the number of distinct laser angles realizable by the measurement set-up. This means that this Optical PUF only has polynomially many challenges. Similar considerations also apply to the Crossbar PUF [17, 18], which only has quadratically many challenges, too. All these PUFs are excluded by Definition 3.4, but especially the Optical PUF and the Crossbar PUF seem fully secure in practice.

4 Alternative Attack Models

Our discussion in the last section showed that the familiar notion of polynomial resources and the usual asymptotic treatment of mathematical cryptography cannot be transferred to PUFs easily. We therefore work out an alternative treatment in this paper, which is based on concrete time bounds. We start with semi-formal models in this section, and provide a more formal version later.

4.1 Semi-Formal Models for Strong PUFs

We start by some fundamentals and some notation for PUFs.

Specification 4.1 (SEMI-FORMAL SPECIFICATION OF PUFS**)** *A PUF is a physical system S that maps stimuli or challenges C_i to responses R_{C_i}. The set of all*

[1] In the original text this sentence is placed after the definition.

possible challenges of S is denoted as $\mathbf{C}_S$*, and the set of all possible responses as* $\mathbf{R}_S$*. Without loss of generality,* $\mathbf{C}_S$ *and* $\mathbf{R}_S$ *are assumed to be finite subsets of* $\{0,1\}^*$*. By its challenge-response behavior, S implements a function* F_S *with*

$$F_S: \quad \mathbf{C}_S \rightarrow \mathbf{R}_S, \quad C \mapsto R_C.$$

We further assume that in each PUF, the responses are notably influenced by fabrication variations beyond the control of the manufacturer (a fact that distinguishes PUFs from purely digital systems based on secret binary keys).

We suggest that apart from these basic requirements, no further security features should be required from a (plain) PUF. In our opinion, such additional security features should strictly be associated with special subnotions derived from PUFs, such as Strong PUFs, Key Obfuscating PUFs or Physically Obfuscated Keys, etc.

We will now present a first, semi-formal security model for Strong PUFs:

Specification 4.2 (SEMI-FORMAL SPECIFICATION OF STRONG PUFS) *Let S be a PUF according to Specification 4.1. S is called a* STRONG $(t_L, t_A, t_P, q, \varepsilon)$-PUF *if no cryptographic adversary Eve limited by the current state of technology will succeed in the following experiment with a probability of at least* ε*.*

SecExp (S, t_L, t_A, t_P, q)*:*

PHASE 1: LEARNING. *Eve is given a time period* t_L *for learning the PUF S. Within that period, she is given physical access to S at adaptively chosen points in time, and for time periods of adaptive length. The sum of all time periods for which she had access to S must not excel* t_A*. Further, Eve can adaptively query an oracle* $\mathcal{O}_{F_S}$ *for the function* F_S *at most q times. After the end of the learning phase, Eve cannot access S or* $\mathcal{O}_{F_S}$ *any more.*

PHASE 2: PREDICTION. *A challenge* C_0 *is chosen uniformly at random from the set* $\mathbf{C}_S$*, and is given to Eve. Within time* t_P*, she must produce an output* V_{Eve}*.*

Thereby the experiment is called successful if $V_{Eve} = R_{C_0}$*. The probability* ε *is taken over the uniformly random choice of the challenge* C_0*, and the random choices or procedures that Eve might employ during Phase 1 and 2.*

The specification models real application scenarios relatively closely. Typically, Eve will have a relatively long "learning period" t_L in practice. During this phase, she may gather information about the PUF in several ways: (i) She can obtain standard CRPs, for example through protocol eavesdropping, via direct physical access to the PUF, or remotely (e.g. by a virus in the embedding system). These possibilities are comprehensively included via the adaptive oracle access and the physical access period t_A that we grant Eve. (ii) Eve may attempt arbitrary measurements (beyond mere CRP determination) on the PUF, including measurement of internal system parameters and invasive probing. This possibility is included in our model through the physical access time t_A. Note that Eve will often not be able to execute her physical access at adaptively chosen points in time, or for periods of adaptive

time length. But specifying our model in this way includes worst-case scenarios, and puts us on the safe side.

In practice, t_L is typically relatively long, and is only limited by the lifetime of the device embedding the PUF and/or the relevance of the data that was encrypted with a key derived from the PUF. Contrary to that, the physical access time t_A is usually short and costly. This motivates a distinction between these two parameters in our model.

The further distinction between t_L and t_P, on the other hand, is not relevant for all applications of Strong PUFs, but plays a role in many of them. In order to obtain definitions with maximal generality, we opted to include it in our model. To illustrate this point, let us consider two typical applications of Strong PUFs, namely key establishment and identification. In key establishment, the main security requirement is that Eve will not be able to predict the responses R_{C_i} that were used to derive a key between the cryptographic players. Usually no distinction between t_L and t_P is necessary here—we are only interested in the sum $t_L + t_P$ of the two values, and hope for the sake of security that $t_L + t_P$ is impractically large. In PUF-based identification, however, an adversarial attack strategy that leads to very long prediction times t_P is worthless. It can be countered in practice through measuring the response time of the identifying party. In other words, Eve's attacks on PUF-based identification protocols are only successful if they deliver the R_{C_i} fast.

Spec. 4.2 provides a workable model for Strong PUFs. The definition is non-asymptotic, whence it allows statements about concrete PUF instances. For example, as Machine Learning results show [19, 20], we can make the following statements:[2]

- A 64-bit Arbiter PUF is no (0.6 sec., 0 sec., 0.001 sec., 18050, 0.99)-Strong PUF.
- A 64-bit Arbiter PUF that produces CRPs at a 1 MHz frequency is no (0.6 sec., 0.01805 sec., 0.001 sec., 0, 0.99)-Strong PUF.

The formalism can also be used to make positive statements, not only negations:

- Assuming that its read-out speed cannot be accelerated[3], a Crossbar PUF of size $10^5 \times 10^5$ and read-out speed of 100 bits/sec. is a (t_L, 10^7 sec., t_P, 0, 0.6)-Strong PUF for arbitrary values of t_L and t_P.
- Assuming that its read-out speed cannot be accelerated, a Crossbar PUF of size $10^5 \times 10^5$ and read-out speed of 100 bits/sec. is a (t_L, 0, t_P, 10^9, 0.6)-Strong PUF for arbitrary values of t_L and t_P.

Specification 4.2 also has its limitations. Most importantly: How do we model Eve? Since we allow Eve arbitrary physical actions, a standard Turing machine is insufficient. This lack of a formal model leads to two problems. Firstly, in a strict

[2] The statements follow from machine learning experiments based simulation data, which were reported in Table 1 of [19]. They show that a 64-bit Arbiter PUF can be broken (in simulations) with the respective parameters in terms of learning times, access times, prediction times, CRPs and prediction rates.

[3] It is argued in [17] in all detail that such an acceleration is indeed practically impossible if the crossbar's design is chosen appropriately.

sense, we do not know over which set we quantify when we state in Specification 4.2 that *"...no cryptographic adversary Eve limited by the current state of technology will succeed in the following experiment..."*. This logical problem is awkward. But it could perhaps be acceptable under the following provisions: (i) Specification 4.2 is not understood as a formal definition, but as a semi-formal specification. (ii) The main purpose of Spec. 4.2 is to put down a precise, but not overly technical description of the essence of Strong PUFs, which can be used as a common basis by all communities involved in PUF research. The second problem that results from the lacking model for Eve is perhaps more severe, at least for theoreticians. Without a formal model for Eve, we cannot perform reductionist security proofs.

In order to resolve this dilemma, we could either introduce a new computational model, which captures arbitrary physical actions (some sort of "*Physical Turing Machine*"). But this seems very intricate. Alternatively, we may restrict the attack model; this route is taken in the rest of the paper.

4.2 The Digital Attack Model

In the *digital attack model*, we follow the basic adversarial model that was put down in Specification 4.2, with one exception: We do not grant Eve direct physical access to the PUF S, and do not allow arbitrary physical measurements on S. Instead, we restrict her to the measurement of CRPs of the PUF. This restriction is not as unrealistic as it may seem: The natural tamper sensitivity of many PUFs enforces this setting by itself. If a PUF is embedded in a device, separating it from the device to make arbitrary measurements will often be impossible.

The advantage of the digital model is that Eve can be formalized by a standard probabilistic Turing machine with an oracle that provides her with CRPs of the PUF S, or, more precisely, with an oracle for the function F_S. This will allow us to carry over reductionist techniques from the security proofs of mathematical cryptography to PUFs.

Let us now define what it means to break the security properties of a Strong PUF in the digital model.

Definition 4.3 (BREAKING STRONG PUFS IN THE DIGITAL ATTACK MODEL) *Let S be a PUF. Let an adversary $\mathscr{A}$ be given by a tuple $(\mathscr{L},\mathscr{M})$, where $\mathscr{L}$ is a probabilistic oracle Turing machine, and $\mathscr{M}$ is a probabilistic Turing machine. We say that $\mathscr{A}$ (t_L, t_P, q, ε)-*BREAKS S AS A STRONG PUF IN THE DIGITAL ATTACK MODEL *if $\mathscr{A}$ succeeds in the following security experiment with a probability of at least ε:*

SecExp (S, t_L, t_P, q)*:*

PHASE 1: LEARNING. *$\mathscr{L}$ is provided with an oracle $\mathscr{O}_{F_S}$ for the function F_S, and is started with an empty input. It may make at most q adaptive queries to $\mathscr{O}_{F_S}$. After at most t_L Turing steps, it outputs a string z and halts.*

PHASE 2: PREDICTION. *A challenge C_0 is chosen uniformly at random from the set $\mathbf{C}_S$, and $\mathscr{M}$ is started with input (z, C_0). Within t_P Turing steps, $\mathscr{M}$ must outputs a string V_{ADV} and halts.*

Thereby the experiment is called successful if $V_{ADV} = R_{C_0}$. The probability ε is taken over the uniformly random choice of the challenge C_0, and the random coins that $\mathscr{L}$ and $\mathscr{M}$ might employ during Phase 1 and 2.

5 Identification Schemes Based on Strong PUFs

We will now work towards a security proof of PUF-based identification schemes in the digital model. We start by defining the concept of a PUF-based identification scheme.

5.1 PUF-based Identification Schemes

Roughly speaking, a PUF-based identification scheme is a protocol where one party $\mathscr{P}$, known as prover, tries to convince another party $\mathscr{V}$, known as verifier, of its identity. The prover possesses a PUF that he can query at will. The protocol should both assert the identity of the prover and the physical availability of the PUF. More precisely, a PUF-based identification scheme is defined as a tuple $(\mathscr{K}, \mathscr{P}, \mathscr{V})$:

Definition 5.1 (PUF-BASED IDENTIFICATION SCHEME**)** *Let S be a PUF. An identification scheme based on S is a tuple $(\mathscr{K}, \mathscr{P}, \mathscr{V})$, where $\mathscr{K}$ is a probabilistic oracle Turing machine, $\mathscr{P}$ is a probabilistic interactive oracle Turing machine and $\mathscr{V}$ is a probabilistic interactive Turing machine, which together fulfill the following properties:*

INITIALIZATION. *On input 1^k, and provided with an oracle $\mathscr{O}_{F_S}$ for the function F_S, the instantiation algorithm $\mathscr{K}$ returns a string x_{in}.*

INTERACTIVE IDENTIFICATION. *In the identification process, $\mathscr{P}$ and $\mathscr{V}$ execute a joint interactive computation, where 1^k is their joint input, $\mathscr{P}$ is provided with an oracle $\mathscr{O}$, and $\mathscr{V}$ gets a private input x. At the end of the computation, $\mathscr{V}$ outputs a bit $d \in \{0,1\}$.*

COMPLETENESS CONDITION. *We require that if in the identification process, $\mathscr{P}$ is run with oracle $\mathscr{O}_{F_S}$, and $\mathscr{V}$ is run with x_{in} as private input, the output of $\mathscr{V}$ at the end of the interactive computation is "1" with probability 1.*

Let us consider the following "canonical" PUF-based identification scheme, illustrated in Figure 1. In a setup phase, the verifier $\mathscr{V}$ chooses a set of k random challenges (where k denotes a security parameter) $C_1, \ldots, C_k$ and measures the PUF response for each challenge. $\mathscr{V}$ stores the set of all chosen challenge-response pairs as private data. Subsequently, the device is given to the prover $\mathscr{P}$. The interactive

identification protocol starts when $\mathscr{P}$ presents its device to a reader: the verifier $\mathscr{V}$ sends the challenges $C_1,\ldots,C_k$ to the prover, who answers with responses $V_1,\ldots,V_k$ which are derived from PUF measurements with challenges $C_1,\ldots,C_k$. $\mathscr{V}$ accepts if all responses match the pre-recorded responses $R_{C_1},\ldots,R_{C_k}$ during initialization.

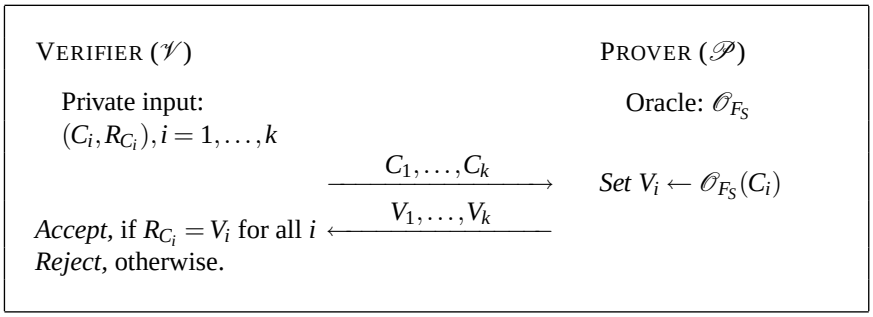

Fig. 1 Canonical identification scheme based on a PUF S

More formally, we define the canonical identification scheme based on a PUF S as the tuple $(\mathscr{K},\mathscr{P},\mathscr{V})$, where the algorithms $(\mathscr{K},\mathscr{P},\mathscr{V})$ implement the above protocol. In particular:

- $\mathscr{K}$ takes as input 1^k and as oracle $\mathscr{O}_{F_S}$. It chooses $C_1,\ldots,C_k$ uniformly at random from the set $\mathbf{C}_S$, and produces as output $x_{in}=(C_1,R_{C_1},\ldots,C_k,R_{C_k})$.
- $\mathscr{V}$ gets the public input 1^k and the private input $x_{in}=(C_1,R_{C_1},\ldots,C_k,R_{C_k})$. It sends $C_1,\ldots,C_k$ to $\mathscr{P}$. Subsequently, it receives values $V_1,\ldots,V_k$ from $\mathscr{P}$, and outputs "1" if and only if

$$V_i = R_{C_i} \quad \text{for all } i=1,\ldots,k.$$

- $\mathscr{P}$ gets as public input 1^k and as oracle $\mathscr{O}_{F_S}$. Upon receiving values $C_1,\ldots,C_k$, it queries $\mathscr{O}_{F_S}$ for the values $V_1=F_S(C_1),\ldots,V_k=F_S(C_k)$, and sends the oracle responses to $\mathscr{V}$.

5.2 Security of PUF-based Identification in the Digital Attack Model

We now state what it means to break a PUF-based identification scheme in the digital attack model. We closely follow the IMP-CA security notion of traditional identification schemes [1]. Thereby, the adversary's goal is to impersonate the prover, that is to make the verifier accept, despite he has no access to the PUF. More precisely, the definition consists of two phases: a *learning* and an *impersonation* phase. In the learning phase, the adversary has access to an oracle $\mathscr{O}_{F_S}$ in order to collect PUF

measurements up to a certain maximum number. Furthermore, the adversary is allowed to play a cheating verifier which can interact with an honest prover for an arbitrary number of independent protocol runs. In the impersonation phase, the adversary tries to impersonate the prover such that the verifier accepts the false proof.

Definition 5.2 *Let S be a PUF, and let $ID_S = (\mathscr{K}, \mathscr{P}, \mathscr{V})$ be an identification scheme based on S. Let an adversary $\mathscr{A}$ be a tuple $(\mathscr{V}^*, \mathscr{P}^*)$, where $\mathscr{V}^*$ is a probabilistic oracle Turing machine, and $\mathscr{P}^*$ is a probabilistic Turing machine. We say that $\mathscr{A}$ $(t_L, t_P, q, r, \varepsilon)$-*BREAKS ID_S FOR THE SECURITY PARAMETER *k if $\mathscr{A}$ succeeds in the following security experiment with a probability of at least ε:*

SecExp (S, t_L, t_P, q, r, k)*:*

PHASE 1: INITIALIZATION. *$\mathscr{K}$ is run on input 1^k and produces an output x_{in}.*

PHASE 2: LEARNING. *$\mathscr{V}^*$ is provided with an oracle $\mathscr{O}_{F_S}$ for the function F_S, and is started with input 1^k. It may make at most q adaptive queries to $\mathscr{O}_{F_S}$. Furthermore, it may interact at most r times with the honest prover $\mathscr{P}$, instantiated with a fresh random tape, whereby $\mathscr{P}$ gets $\mathscr{O}_{F_S}$ as oracle and 1^k as input at each of these interactions. After at most t_L Turing steps, $\mathscr{V}^*$ must output a string z.*

PHASE 3: IMPERSONATION. *$\mathscr{P}^*$ is provided with the private input z. $\mathscr{V}$ is provided with the private input x_{in}. Both get as joint input 1^k, and execute a joint computation. Within t_P Turing steps, $\mathscr{V}$ outputs a bit $d \in \{0,1\}$.*

We say that the experiment was successful if $\mathscr{V}$ outputs "1" at the end of Phase 3. The probability ε is taken over the random coins that $\mathscr{K}, \mathscr{V}, \mathscr{P}, \mathscr{V}^*, \mathscr{P}^*$ employ during Phases 1 to 3.

We will now perform a reductionist security proof for the canonical PUF-based identification scheme. The following statement informally says that if S is a Strong PUF, then the canonical identification scheme based on S is secure.

Theorem 1. *Let S be a PUF. Then there is a generic black-box reduction that constructs from any adversary $\mathscr{A} = (\mathscr{V}^*, \mathscr{P}^*)$, which $(t_L, t_P, q, r, \varepsilon)$-breaks the canonical identification scheme based on S for the security parameter k, another adversary $\mathscr{A}' = (\mathscr{L}, \mathscr{M})$, which $(t_L + c \cdot \lceil k/\varepsilon \rceil, \lceil k/\varepsilon \rceil (t_P + c \cdot k), q + kr + \lceil k/\varepsilon \rceil, 0.6\sqrt[k]{\varepsilon}/k)$ breaks S as a Strong PUF. Thereby c is a small constant independent of k.*

Proof. In the following, we show how to build an adversary $\mathscr{A}' = (\mathscr{L}, \mathscr{M})$ that predicts the response to a given challenge by running black-box simulations of $\mathscr{A} = (\mathscr{V}^*, \mathscr{P}^*)$.

More precisely, $\mathscr{L}$ runs a black-box simulation of $\mathscr{V}^*$. Whenever $\mathscr{V}^*$ makes a query to $\mathscr{O}_{F_S}$, $\mathscr{L}$ simulates this query by using his oracle $\mathscr{O}_{F_S}$. $\mathscr{L}$ keeps track of all oracle queries and their responses in a list *crp* to avoid duplicate oracle queries. Whenever $\mathscr{V}^*$ engages in a protocol run with the prover, $\mathscr{L}$ simulates this interaction as follows: upon receipt of a list of k challenges $(C_1, \ldots, C_k)$, $\mathscr{L}$ creates a corresponding list of PUF responses $(R_{C_1}, \ldots, R_{C_k})$, either by looking up the result in *crp* or querying $\mathscr{O}_{F_S}$. Once $\mathscr{V}^*$ stops with output z, $\mathscr{L}$ proceeds to draw $\ell = \lceil k/\varepsilon \rceil$

further (previously unused) challenges randomly from the set of challenges and obtains their answers by querying $\mathscr{O}_{F_S}$; all challenges and responses collected in this last step are collected in a list $CR = (\hat{C}_1, \hat{R}_1, \ldots, \hat{C}_\ell, \hat{R}_\ell)$. Subsequently $\mathscr{L}$ halts and outputs (z, CR).

On receiving a challenge C_1, $\mathscr{M}$ performs the following operations:

1. $\mathscr{M}$ selects uniformly at random a position k_0 with $1 \leq k_0 \leq k$ and constructs a list of k challenges $(\bar{C}_1, \ldots, \bar{C}_k)$ as follows: he sets $\bar{C}_{k_0} = C_1$ and $\bar{C}_i = \hat{C}_i$ for $1 \leq i \leq k_0 - 1$; furthermore he sets all $\bar{C}_i$ with $k_0 + 1 \leq i \leq k$ to random challenges from $\mathbf{C}_S$.
2. $\mathscr{M}$ runs $\mathscr{P}^*$ on $(\bar{C}_1, \ldots, \bar{C}_k)$ and input z to obtain $(\bar{R}_1, \ldots, \bar{R}_k)$.
3. If $\bar{R}_i = \hat{R}_i$ for $1 \leq i \leq k_0 - 1$, algorithm $\mathscr{M}$ outputs $\bar{R}_{k_0}$ and stops. Otherwise, $\mathscr{M}$ deletes the first (used) $k_0 - 1$ entries of the list CR and re-starts the operation at step 1. After $m = \lceil k/\varepsilon \rceil$ unsuccessful runs, $\mathscr{M}$ stops and fails.

We denote by A_i the probability that $\mathscr{P}^*$ (when run in the game of Definition 5.2) outputs the correct response for the i-th challenge it is given. We thus have $\text{Prob}[\mathscr{P}^* \text{ succeeds}] = \text{Prob}[\bigcap_{i=1,\ldots,k} A_i] > \varepsilon$. We can write $\text{Prob}[\bigcap_{i=1,\ldots,k} A_i]$ as

$$\text{Prob}[A_1]\,\text{Prob}[A_2 \,|\, A_1]\,\text{Prob}[A_3 \,|\, A_2 \cap A_1] \ldots \text{Prob}[A_k \,|\, A_{k-1} \cap \ldots \cap A_1].$$

Since $\text{Prob}[\bigcap_{i=1,\ldots,k} A_i] > \varepsilon$, we know that one of the factors in the above product must be larger than $\sqrt[k]{\varepsilon}$. Thus, there exists a position $1 \leq \bar{k} \leq k$ in which $\mathscr{P}^*$ succeeds with a higher probability, under the condition that the algorithm has predicted all prior challenges correctly. The reduction attempts to exploit this fact. It guesses this position $\bar{k}$. Then, it outputs the response of $\mathscr{P}^*$ for the $\bar{k}$-th challenge, but only in case $\mathscr{P}^*$ has predicted all previous challenges correctly. Otherwise, this (sub-)run of $\mathscr{M}$ fails, and a new run is started, up to $m = \lceil k/\varepsilon \rceil$ overall iterations.

The probability that $\mathscr{M}$ succeeds to guess this position $\bar{k}$ in one iteration is $1/k$; the probability that $\mathscr{M}$ outputs a correct guess in this round is $\text{Prob}[A_{\bar{k}} \,|\, A_{\bar{k}-1} \cap \ldots \cap A_1] \geq \sqrt[k]{\varepsilon}$, since the reduction is constructed in a way that it outputs a guess only if $\mathscr{P}^*$ predicts all challenges $\bar{C}_1, \ldots, \bar{C}_{\bar{k}}$ correctly. Due to the independence of successive runs of $\mathscr{P}^*$, we can estimate the overall success probability of $\mathscr{M}$ as

$$\begin{aligned} \text{Prob}[\mathscr{M} \text{ succeeds}] &\geq 1/k(1-(1-\varepsilon)^{k/\varepsilon})\sqrt[k]{\varepsilon} \\ &\geq 1/k(1-(1/e)^k)\sqrt[k]{\varepsilon} \\ &\geq 0.6\sqrt[k]{\varepsilon}/k. \end{aligned}$$

The bounds on the run times of $\mathscr{V}^*$ and $\mathscr{P}^*$ can easily be obtained by observing that the simulation requires an overhead that scales linearly in the security parameter k. The precise scaling constant c is dependent on the machine model, and is independent of k. Furthermore, $\mathscr{V}^*$ makes at most $q + kr + \lceil k/\varepsilon \rceil$ oracle queries. This proves the theorem.

6 Conclusions

We investigated the formal foundations and applications of Strong Physical Unclonable Functions. The problem of defining PUFs and Strong PUFs in a formally sound way turns out to be complicated. One reason for the occurring obstacles is that PUFs are a hybrid between physics and computer science. Some of their properties, such as their unclonability or the dependence of their output on uncontrollable manufacturing variations, can hardly be expressed in a formalism based on standard Turing machines. On the other hand, some other central features of Strong PUFs—such as their unpredictability, even if many CRPs are known—are closely related to computational complexity. Expressing them formally requires some sort of computational model. Finally, a purely information-theoretic approach to PUF-security is not going to work for all Strong PUFs: Several electrical Strong PUF candidates contain only a relatively small amount of relevant structural information.

We made the following contributions. We started by analyzing existing definitions of Physical One-Way Functions, Physical Random Functions and Physical Unclonable Functions, and noted some interesting aspects in these definitions. We subsequently proposed new semi-formal specifications for Strong PUFs. They have some limitations from a strictly formal point of view, and do not enable reductionist proofs. But they are intuitive and not overly technical, and also specify an adversarial model and its security relevant parameters relatively exactly. The specifications also have the asset of being non-asymptotic, meaning that they can be applied directly to concrete PUF-instances.

Next, we introduced a restricted adversarial model, the *digital attack model*, and gave a security definition for breaking Strong PUFs, which eventually enabled reductionist proofs. In principle, it is based on the adversarial scenario of the above informal specifications. But it restricts the adversary's measurements on the PUF to mere CRP determination. This constraint allowed to model the adversary by oracle Turing machines, and made classical reductionist techniques applicable. Finally, we showed that the security of the "canonical" PUF identification scheme can be provably based on the security of the underlying Strong PUF without any complexity theoretic assumptions.

References

1. Mihir Bellare and Adriana Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *Advances in Cryptology (CRYPTO 2002), Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 2002.
2. Qingqing Chen, György Csaba, Xueming Ju, Srinivas Bangalore Natarajan, Paolo Lugli, Martin Stutzmann, Ulf Schlichtmann, and Ulrich Rührmair. Analog circuits for physical cryptography. In *International Symposium on Integrated Circuits (ISIC)*, 2009.
3. György Csaba, Xueming Ju, Zhiqian Ma, Qingqing Chen, Wolfgang Porod, Jürgen Schmidhuber, Ulf Schlichtmann, Paolo Lugli, and Ulrich Rührmair. Application of mismatched cellular nonlinear networks for physical cryptography. In *12th IEEE CNNA - International workshop on Cellular Nanoscale Networks and their Applications*, 2010.
4. Srinivas Devadas G. Edward Suh. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the Design Automation Conference, DAC 2007*, pages 9–14.
5. B. Gassend, D. Lim, D. Clarke, M. v. Dijk, and S. Devadas. Identification and authentication of integrated circuits. *Concurrency and Computation: Practice & Experience*, 16(11):1077–1098, 2004.
6. Blaise L.P. Gassend. Physical random functions. Master thesis, Massachusetts Institute of Technology, February 2003.
7. Oded Goldreich. *Foundations of Cryptography*, volume 1, Basic Tools. Cambridge University Press, 2001.
8. Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, and Pim Tuyls. FPGA intrinsic PUFs and their use for IP protection. In *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 63–80. Springer, 2007.
9. Ryan Helinski, Dhruva Acharyya, and Jim Plusquellic. A physical unclonable function defined using power distribution system equivalent resistance variations. In *Proceedings of the 46th Design Automation Conference (DAC 2009)*, pages 676–681, 2009.
10. Sandeep S. Kumar, Jorge Guajardo, Roel Maes, Geert Jan Schrijen, and Pim Tuyls. The Butterfly PUF: Protecting IP on every FPGA. In *International Symposium on Hardware-Oriented Security and Trust (HOST 2008)*, pages 67–70, 2008.
11. J. W. Lee, Daihyun Lim, Blaise Gassend, G. Edward Suh, Marten van Dijk, and Srinivas Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *Proceedings of the IEEE VLSI Circuits Symposium*, pages 176–179. IEEE, 2004.
12. Daihyun Lim. Extracting secret keys from integrated circuits. Master's thesis, Massachusetts Institute of Technology, 2004.
13. Daihyun Lim, J. W. Lee, Blaise Gassend, G. Edward Suh, Marten van Dijk, and Srinivas Devadas. Extracting secret keys from integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(10):1200–1205, October 2005.
14. Mehrdad Majzoobi, Farinaz Koushanfar, and Miodrag Potkonjak. Lightweight secure PUFs. In *International Conference on Computer-Aided Design (ICCAD'08)*, pages 670–673, 2008.
15. Ravikanth Srinivasa Pappu. *Physical One-Way Functions*. Phd thesis, Massachusetts Institut of Technology, March 2001.
16. Ravikanth Srinivasa Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical one-way functions. *Science*, 297(5589):2026–2030, September 2002.
17. Ulrich Rührmair, Christian Jaeger, Matthias Bator, Martin Stutzmann, Paolo Lugli, and György Csaba. Applications of high-capacity crossbar memories in cryptography. *IEEE Transactions on Nanotechnology*, 2010, to appear.
18. Ulrich Rührmair, Christian Jaeger, Christian Hilgers, Michael Algasinger, György Csaba, and Martin Stutzmann. Security applications of diodes with unique current-voltage characteristics. In *Financial Cryptography (FC 2010)*.

19. Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Vera Stoyanova, and Jürgen Schmidhuber. Machine learning attacks on physical unclonable functions, submitted. 2009.
20. Ulrich Rührmair, Jan Sölter, and Frank Sehnke. On the foundations of physical unclonable functions. Technical Report 227, IACR Cryptology E-print Archive, 2009.
21. Pim Tuyls, Geert Jan Schrijen, Boris Škorić, Jan van Geloven, Nynke Verhaegh, and Rob Wolters. Read-proof hardware from protective coatings. In *Cryptographic Hardware and Embedded Systems (CHES 2006), Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 369–383. Springer, 2006.