

On the Foundations of Physical Unclonable Functions

Ulrich Rührmair

Jan Sölter

Frank Sehnke

June 10, 2009

Abstract

We investigate the foundations of Physical Unclonable Functions from several perspectives. Firstly, we discuss formal and conceptual issues in the various current definitions of PUFs. As we argue, they have the effect that many PUF candidates formally meet no existing definition. Next, we present alternative definitions and a new formalism. It avoids asymptotic concepts like polynomial time, but is based on concrete time bounds and on the concept of a security experiment. The formalism splits the notion of a PUF into two new notions, Strong t -PUFs and Obfuscating t -PUFs.

Then, we provide a comparative analysis between the existing definitions and our new notions, by classifying existing PUF implementations with respect to them. In this process, we use several new and unpublished machine learning results. The outcome of this comparative classification is that our definitions seem to match the current PUF landscape well, perhaps better than previous definitions. Finally, we analyze the security and practicality features of Strong and Obfuscating t -PUFs in concrete applications, obtaining further justification for the split into two notions.

1 Introduction

Physical One-Way Functions (POWFs), Physical Random Functions (PRFs), and Physical Unclonable Functions (PUFs) are emerging as a new type of cryptographic primitive [1, 2, 3, 5, 6, 7, 10, 4]. While the two former terms have been coined first, today mainly the expression Physical Unclonable Function or PUF is in use. In a nutshell, a PUF is a mathematical function that is derived from the behaviour of a complex physical object or device. The object can be excited with many possible stimuli C_i , upon which it reacts with corresponding responses R_i . This allows to regard the object's behavior as a function that maps stimuli C_i to responses R_i . Typical applications of PUFs lie in security tasks such as tamper protection, intellectual property protection, read-proof memory, and key obfuscation. They can also be used for classical cryptographic protocols such as key distribution [1] or bit commitment [2].

Despite their broad application spectrum, there are yet no definitions that capture the notion of a PUF consistently and in a contradiction-free manner. Most existing formalizations exhibit problems on the formal and/or on the conceptual side, as we will argue later. Furthermore, it seems that currently more than one concept or notion goes under the term of a "PUF": On the one hand, structures with a highly complex input-output behavior and very many challenges, such as the optical PUF of [1]. On the other hand, structures that have essentially one challenge [6, 10], and which are mainly used for key obfuscation applications or as so-called "POKs" [3].

We take this situation as a reason to investigate the foundations of PUFs further. We start by a thorough formal analysis of currently existing PUF-definitions (section 2). Some problematic aspects are revealed, which are in part caused by the application of concepts like polynomial time and negligible information in the context of PUFs. Subsequently, we propose a new formalism and new definitions (sections 3 and 4). We suggest that the current concept of a PUF should be split into two distinct notions: Strong PUFs and Obfuscating PUFs. Our definitions avoid asymptotic notions, and are based on concrete time bounds and the concept of a security experiment. They build upon several previously existing concepts and definitions [2, 3, 10].

We then go on to classify current PUF implementations with respect to our two new definitions, and also with respect to the previously existing definitions (section 5). In this process, we use several new and unpublished machine learning results that were recently obtained in our group. They show that not only the arbiter PUF, but also improved variants (XOR Arbiter [5, 7], Lightweight PUF [12] and Feed-Forward Arbiter [5]) can all be broken by ML techniques. The resulting classification suggests that our definitions can map the current PUF landscape well, perhaps even more consistently than previous approaches.

Finally, we argue that also from an application perspective, it makes sense to maintain the suggested split into two new notions (section 6). To that end, we analyze the differing security and practicality features that occur in the application of strong and obfuscating PUFs for entity identification. In section 7, we argue that a general emulation of strong PUFs is possible on the basis of obfuscating PUFs, but at the cost of reduced security and practicality. Once more, this strengthens the case for our distinction between strong and obfuscating PUFs. We conclude the publication by a summary in section 8.

2 Problems with Existing Definitions of PUFs

Let us analyze the formal definitions/specifications of PUFs that have been published up to date. For the convenience of the reader, these definitions are provided in the Appendices A, B and C. *Please note* that each of these definitions possesses its own individual merits, and that they are significant pieces of scientific work. Our discussion merely would like to point at the non-trivial obstacles that naturally and necessarily occur in the formulation of a consistent and workable definition of PUFs.

2.1 Physical One-Way Functions

We start with the definition of physical one-way functions [2], which is given in Appendix A. It certainly owns the merit of being the first formalization in the field. But there are some problematic aspects, both on the formal/logical and on the conceptual side, which we list below.

Asymptotic concepts vs. finite function. The definition employs the concept of polynomial resources and of negligible probability, which are familiar from the formalization of mathematical cryptography, for example from the definition of (mathematical) one-way functions [17]. However, these concepts can only be applied in an asymptotic framework, that is, when they are considered for functions with an infinite domain, or for an infinite family of finite functions. If the concepts are applied in finite contexts, logical contradiction arise.

In particular, Definition A.2 considers only one single function f with a finite domain. This has the consequence that *no* function at all can formally fulfill Definition A.2, as the following logical argument shows: Let f be an arbitrary mathematical functions, and suppose, for the sake of contradiction, that f meets Definition A.2. As the definition considers only finite functions, f must

be finite, with a finite domain and range. This implies that an algorithm A_f with the following properties can be constructed: (i) A_f incorporates a full look-up table for f . (ii) A_f inverts f by exhaustive search through the domain of the look-up table. (iii) A_f works within a constant time bound (which is essentially the time it takes to browse the look-up table). However, the existence of A_f violates item 2 of the Def. A.2, whence f does not meet the definition, contradiction. This means that no Physical One-Way Function in the sense of Definition A.2 exists.

Functions with small ranges are excluded. Def. A.2 excludes functions with small ranges, for example with the binary output range $\{0, 1\}$. Such functions can be inverted in the sense of item 2 of the definition, as *some* pre-image for one of the two possible function values can be found easily. Many prominent examples of electrical PUFs (arbiter PUF [7], ring oscillator PUF [7], etc.) belong to this class of PUFs, and are excluded by the definition.

Non-invertability is not the relevant security property. Another conceptual argument is that the non-invertability of f is not required in order to make the standard applications of physical one-way functions secure. As an example, consider the two main applications proposed in [1], the forgery-proof labeling of bankcards and key distribution. Speaking in the language of [2] and Notation A.1, they merely require that an adversary without access to the physical system Σ cannot give the correct function value $f(X, P_z)$ for a randomly chosen z , even if he has had previous access to the system Σ for some time.

2.2 Physical Random Functions

Let us now elaborate on the definition of Physical Random Functions (see Appendix B) for comparison. It is certainly very compact and intuitively appealing, and also includes some sort of asymptotic treatment: The notions polynomial and negligible are taken to be relative to the size of the system (see Appendix B). But the definition nevertheless touches upon some problematic issues.

PUFs with polynomially many challenges are excluded. Definition B.1 allows an adversary to measure polynomially many challenge response pairs (CRPs). This has the consequence that several of the most prominent examples of PUFs will not 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 subsequently use the table for correct PUF prediction. This applies to the ring oscillator PUF proposed in [7], which has only a quadratic number of challenges. It surprisingly holds for the optical PUF of [1], too: Its number of CRPs is directly proportional to the x and y dimensions of the scattering token, multiplied by a constant factor for the finite number of distinct laser angles realizable by the limited measurement set up (this latter number does not grow with the token size). Excluding these PUFs for formal reasons, while especially the optical PUF seems secure by all practical standards, seems problematic.

Information content of physical systems is polynomially bounded. It is worth noting in this context that the amount of information (measured in bits) that can maximally be stored in any physical system S is always polynomially bounded in the size of the system. This can be seen in two ways. Firstly by some simple heuristic: The amount of atoms (or electrons or quarks etc.) in a physical system is polynomially related to the volume. If each of these particles can store a constant number of bits, the information content is polynomial. Even if we speculate that each particle could store a constant (or even a polynomial!) number of bits in its precise relation to each

of the other, polynomially many particles (for example by coupling), the overall number of stored bits would still remain polynomial.

There is, however, also a more fundamental argument to the same end. It has been known for a few years that the generalized second law of thermodynamics can be used to derive bounds on the maximal entropy (or information content) of a finite, isolated system. The two most famous bounds are the Bekenstein bound and the holographic bound by t’Hooft and Susskind [20]. The latter states that for any isolated physical system S which fits into a sphere of radius R , the maximal entropy or information content H_S of S is bounded by

$$H_S \leq \pi c^3 R^2 / \hbar G. \quad (1)$$

Thereby c denotes the speed of light, G is Newton’s constant, and \hbar is the Planck constant. The equation establishes the polynomial upper bound that we sought.

If the maximal information content of any physical system is upper bounded polynomially in its size (volume/area), however, this makes it questionable whether the usual polynomial/super-polynomial distinction is the right measure to characterize PUFs.

2.3 Physical Unclonable Functions

Another characterization of PUFs was given in [10] (see Description C.1 in Appendix C). It is not formally stated as a definition in the original text, whence we term it as a *description*. It has the great benefit of introducing aspects like measurement noise and also tamper sensitivity. But there are a few problematic aspects related to its specifications.

Asymptotic concepts vs. finite function. It seems reasonable to conclude that a PUF implemented by a finite device is indeed a finite function. This means that in Description C.1, the asymptotic concepts of negligible probability and the notion of a finite function are again mixed. By a similar argument as carried out full detail in section 2.1, this has the consequence that formally no physical unclonable function in the sense of Description C.1 can exist.

Functions with small ranges are excluded. Item 2 of the description states that it must be impossible to come up with the response to a random challenge, except with negligible probability. However, most known electrical PUFs only have one single bit as output (e.g. ring oscillator and arbiter PUFs, and also any variants of the arbiter PUF such as feed-forward arbiter, XOR arbiter, or leightweight PUFs). Hence, an adversary can always guess their output with probability $1/2$. This means that all of these PUFs are excluded by the description, which is not desired.

Strong PUFs in the sense of Description C.1 cannot exist. As argued in detail in section 2.2, the information content of an isolated physical system is bounded polynomially in its size (volume/surface area). This means that a strong PUF with an exponential number of challenges in the sense of Description C.1 cannot exist. This is shown by the following argument: Suppose, for the sake of contradiction, that a strong PUF according to Description C.1 exists. As stipulated by the description, such a PUF must possess an exponential number of challenges. At the same time, item 1 of the description states that the responses only give a pairwise negligible information about each other. This implies that the CRPs extract an exponential amount of information from a system that maximally can contain a polynomial amount of information, contradiction. Hence, no strong PUF in the sense of Description C.1 can exist.

Weak PUFs in the sense of Description C.1 are a very restrictive notion. Weak PUFs in the sense of Description C.1 may well exist, but the concept of a weak PUF may turn out to be quite a restrictive notion. As our previous discussion indicates, the only known candidates for a weak PUF are coating PUF and SRAM-based PUFs [6, 10]. Similar to our above discussion, Arbiter PUFs and Ring Oscillator PUFs do not fulfill the requirement that their challenges merely reveal a negligible amount of information about each other, and cannot be weak PUFs.

2.4 Summary

The current definitions of PUFs exhibit problems, which mainly arise from the use of notions like polynomial time or negligible probability in the context of PUFs. Our discussion suggests that these measures should be avoided. They also have problems in formally expressing the unclonability of the system underlying the PUF. In consequence, many current PUF candidates meet none of the existing definitions in a formal manner (see also section 5.1).

3 Strong t -PUFs

This section covers the first of the two new notions suggested in this paper, Strong t -PUFs. We start by some notation.

Notation 3.1 (Measurements). *We use the following notation in order to describe the measurement of an apparatus M on a physical system S : C_i denotes the stimulus or challenge which the measuring apparatus applies to the system. M_{C_i} or $M_{C_i}^S$ denotes the measurement result or the response that is obtained by the apparatus in dependency on C . \mathbf{C}_M denotes the finite set of all possible challenges C_i which M can apply to S .*

Definition 3.2 (Strong t -PUFs). *Let S be a physical system and M be a measuring apparatus, which may be integrated into the system. S is called a STRONG t -PUF or simply a t -PUF with respect to M if the following holds:*

1. *It is practically infeasible for the original manufacturer of S to produce another system S' with*

$$M_C^S = M_C^{S'} \quad \text{for all } C \in \mathbf{C}_M.$$

2. *It is practically infeasible for any cryptographic adversary Eve, who may execute any physical action allowed by the current state of technology and any practically feasible Turing computation, to succeed in the following experiment with a probability greater than 90%:*

- (a) *Eve is given the system S and the measurement apparatus M for a time period t .*
- (b) *Eve is also granted access for time t to a “time-faithful” oracle O , which outputs values M_C^S on input C . Time-faithful means that O produces its output in the same time span that would be required for measuring the response M_C^S on the real system S via use of M .*
- (c) *At the end of the period of length t , Eve must output a physical system S' , and access to O is withdrawn from her.*
- (d) *Subsequently, a measurement parameter C_0 is chosen uniformly at random from the set \mathbf{C}_M , and is given to Eve. After that, she must answer with a numerical value V_{Eve} .*

The experiment is called successful if the following holds:

- (i) $V_{Eve} = M_{C_0}^S$.
- (ii) For all $C \in \mathbf{C}_M$ it holds that
- $$M_C^S = M_C^{S'}$$

Thereby the probability is taken over the uniformly random choice of the challenge C_0 , and the random choices or procedures that Eve might employ during steps 2a to 2d.

Eve's task in the security experiment is depicted schematically in Figure 1. In order to achieve ease of use of our terminology, we will specify a default value for the parameter t .

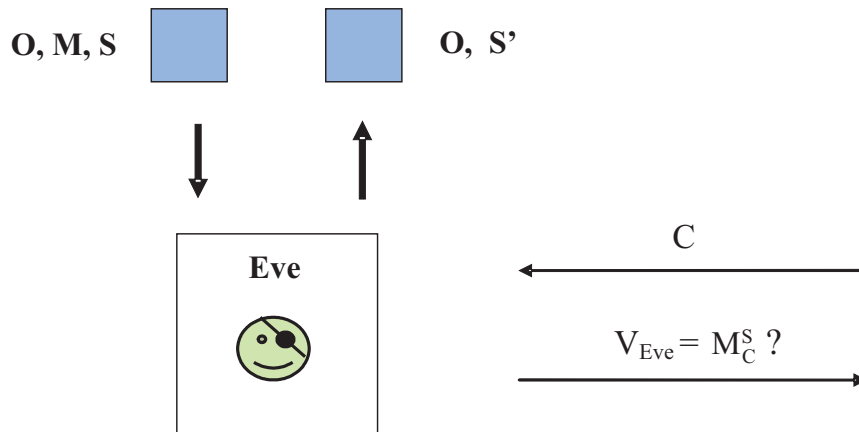


Figure 1: Eve's task in the security experiment of Definition 3.2. She is treated as a black box, and only her input-output behavior is specified.

Notation 3.3 (PUFs). *If S is a strong t -PUF with respect to some measuring apparatus M and with respect to a time period $t \geq 1$ day, then we will often call S simply a STRONG PUF or just a PUF.*

3.1 Discussion

Let us discuss the central features of the definition.

Why S and S' ? The reader may ask why we stipulated that Eve must output a system S' after a time period t . It might seem more suggestive to replace item 1. of Definition 3.2 by the following item 1.':

- 1.' *Eve is given the system S and the measurement apparatus M for a time period t . At the end of that period, Eve must hand back the system S .*

Such a statement assumes that Eve leaves the system unchanged, however, thus making a very strong assumption about her internal behavior. Otherwise, i.e. if we formally denote a strongly altered system with different response values still as S , we run into the logical contradiction that $M_C^S \neq M_C^S$ (for at least some $C \in \mathbf{C}$). The cleanest solution to our taste is to differentiate between the input and output object, and to demand that $M_C^S = M_C^{S'}$ for all $p \in \mathcal{P}_M$. Please note that this also condition also must be met by Eve in practical attacks, if she wants her presence to remain undetected.

Why a success probability of 90%? This allows the use of systems with a binary output (that is, 0 or 1) as strong PUFs, since the chance of guessing a single bit value correctly is always at least $1/2$. Also, it gives credit to the fact that the natural manufacturing variations in certain systems are not big enough as to change 50% of all responses [4]. Choosing the value 90% instead of 75%, say, is to some extent arbitrary.

Purpose of the oracle O . The main reason for including the oracle O in the definition is to distinguish strong PUFs from systems whose security stems from an access restriction of any kind. Such access restrictions are, for example, present in controlled PUFs, or also in coating PUFs or related systems, where the responses cannot be measured from the outside without tampering the structure. In contrast to that, the security of strong PUFs should only lie in the high physical complexity or disorder of the structure underlying the PUF. PUFs which are secure in the face of such oracle attacks are also automatically secure against any side channel attacks.

All statements relative to M . Our definition stresses the central role of a measuring apparatus in connection with strong PUFs. Being a strong PUF is not a property of the system alone, but of the system *and* the method or apparatus by which the system is measured. This also allows us to easily include the unclonability of a strong PUF in the definition.

Secret key based systems are excluded. Please note that by the manufacturer resistance condition in item 1 of the definition, tamper sensitive systems which contain a secret binary key and use this classical key to generate a complex input-output behavior, are excluded from the definition.

Implications of the Definition. The reader can verify easily that several natural properties of PUFs follow from the definition. For example, the set \mathbf{C}_M must have large cardinality, since otherwise Eve could perform a complete read-out and create a full look-up table even for small values of t . Furthermore, Eve must be unable to machine learn the system, or to clone it physically, in the sense that she can output a second system S^* such that $M_C^S = M_C^{S^*}$ for at least 90% of all $C \in \mathbf{C}_M$.

4 Obfuscating t -PUFs

PUFs have also been suggested for a second type of application, which has been termed “key obfuscation” [3, 6]. The idea is to use the disordered, unique internal structure of a PUF as a non-volatile storage of a secret binary key, whose content can hardly be determined through external, invasive attacks due to its irregular and disordered nature. Typical representatives of this class are PUFs based on SRAM-cells [10] and coating PUFs [6]. Please note that these two PUFs are neither strong PUFs in the sense of Definition 3.2, nor Physical One-Way Functions, Physical Random Functions, or strong Physical Unclonable Functions in the sense of the respective Definitions A.2, B.1, C.1. Therefore we introduce *obfuscating PUFs*. The previous notions that come closest to obfuscating PUFs are weak Physical Unclonable Functions (Definition C.1) and Physically Obfuscated Keys [3].

Definition 4.1 (Obfuscating PUFs). *Let S be a physical system, and M be a measuring apparatus with only one measurement parameter, that is, $\mathbf{C}_M = \{C^*\}$. S is called an OBFUSCATING t -PUF for a binary key K_S relative to M if the following holds:*

1. $M_{C^*}^S = K_S$.

2. The value of K_S results, at least in part, from random, uncontrollable manufacturing variations.
3. It is infeasible for Eve to succeed in the following experiment with a probability greater than $(0.9)^{|K_S|}$:
 - (a) Eve is given S and M for a time period t . She is allowed any action on the systems S and M permitted by current technology.
 - (b) At the end of that period, Eve must output a binary value V_{Eve} .
 - (c) The experiment is called successful if $V_{Eve} = K_S$.

Thereby the probability is taken over the random choices or procedures that Eve employed during steps 3a and 3b.

Eve's task is illustrated in Figure 2. In order to achieve an easy notation, we introduce the following notational convention.

Notation 4.2 (Obfuscating PUFs, POKs). *If S is an obfuscating t -PUF with respect to some measuring apparatus M and with respect to a time period $t \geq 1$ day, then we will often call S simply an OBFUSCATING PUF or a POK.*

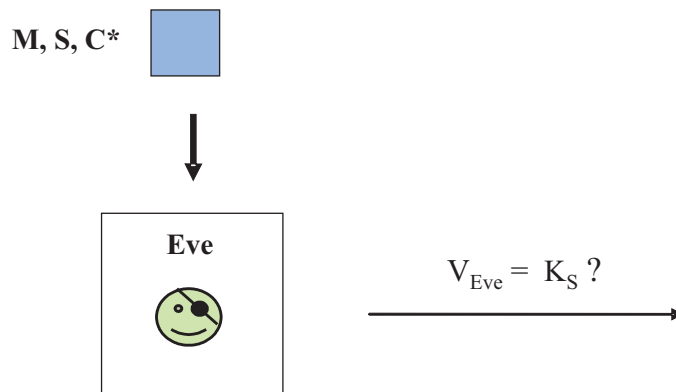


Figure 2: The task that Eve needs to accomplish in order to break the security of an obfuscating PUF

4.1 Discussion

Manufacturer Resistance. We would like to emphasize that we stipulated different types of 'manufacturer resistance' in Definitions 3.2 and 4.1. The requirement expressed in item 1 of Definition 3.2 does not make sense for obfuscating PUFs: Their information content is not necessarily very high. This may allow the manufacturer to fully characterize the system, and/or to extract the obfuscated key K_S during a certain substep of the manufacturing process. Subsequently, he may construct another piece of hardware S' , which simply stores K_S in classical, non-volatile form. This system will behave indistinguishably from the original S , violating the original manufacturer resistance stated in item 1 of 3.2. Therefore another type of manufacturer resistance had to be expressed in Def. 4.1.

Only one challenge. Definition 4.1 stipulates that an obfuscating PUF must only have one measurement parameter. The reason is that in practical applications, the hardware system containing the obfuscating PUF always uses it to derive the same key (or the same small set of keys), and proceeds these keys by further cryptoschemes. This implies that some form of fixed challenge must be hardwired into the system. The definition pays respect to this fact.

Distinction to systems with binary keys. The type of manufacturer resistance expressed in item 2 of the definition also distinguishes obfuscating PUFs from classical, tamper sensitive systems that store a protected, secret key. Such systems do not, and should not, meet the definition.

5 Classification

We will now try to classify the main Physical Unclonable Functions that have been proposed to date with respect to our two new notions. Please note that – similar to classical cryptography – we cannot formally prove that a certain candidate meets one of the definitions, we can only *disprove* it. For brevity, we only sketch the respective arguments.

Optical PUF. Due to its high internal complexity and in lack of a known method to reverse engineer or machine learn it, the optical PUF of [1] is a good candidate for a strong PUF (in the sense of Def. 3.2. Since the measurement apparatus and also the measurement process can be observed by Eve, she can derive the measurement angle(s) and point(s) of incidence that are used for the derivation of a potentially obfuscated key K_S . The optical PUF hence does not seem suitable as obfuscating PUF, as long as it is not integrated.

Optical Integrated PUF. An integrated optical PUF has been proposed in [8] Its internal complexity is substantial, but does not seem to be as outstanding as in the case of the original optical PUF [8]. In lack of a proven method to attack it, it must be regarded as a candidate for a strong PUF. Due to its integrated nature, it is also a candidate for an obfuscating PUF.

Ring Oscillator PUF. A ring oscillator PUF (RO PUF) [7] with n ring oscillators can be fully characterized with $n \cdot \log n$ bits of information (i.e. via a list that sorts the n oscillators in ascending order with respect to their frequencies). If the challenges are chosen carefully, a set of challenge-response-pairs of comparable cardinality suffices in order to derive this list. In any case, the maximal set of all possible CRPs has only cardinality $O(n^2)$, making even a full read-out practically feasible. Thus, the RO PUF is no strong PUF. At the same time, it is a candidate for an obfuscating PUF.

Arbiter PUF. Arbiter PUFs [7] have an exponential number (in the number of stages) of possible challenges, meaning that full read-out of all CRPs is impossible. Nevertheless, machine learning of their behavior has been carried out successfully by standard ML methods (support vector machines [5] and even perceptrons [4]). This means that they are no strong PUFs, but they could be used as obfuscating PUFs.

XOR Arbiter PUF. XOR Arbiter PUFs [5, 7] consist of a number of k independent Arbiter PUFs, each with the same number of stages. The same challenge C_i is applied to all of them, and

their responses R_i^1, \dots, R_i^k are XORed with each other to obtain the overall response R_i .¹ They have been introduced to make the known ML attacks on Arbiter PUFs more difficult. One obvious disadvantage is that their read-out stability decreases exponentially in k . Recently, our group has carried out ML experiments on XOR Arbiter PUFs with 4 XORed arbiters. The methods we applied was logistic regression with Rprop gradient descent. The data samples (training set, test set) were generated by an additive linear delay model. As described in greater detail in Appendix C, we achieved prediction rates of 99 %, meaning that the XOR Arbiter PUF can be fully broken by this type of ML attack. Hence, it is no strong PUF, but a candidate for an obfuscating PUF.

Lightweight PUFs. Lightweight PUFs (LW PUFs) have been introduced in [12]. They are, in fact, a special sort of XOR arbiter PUFs, with the only difference that the wiring of the external challenge bits to the internal challenges that are applied to the k arbiters is non-trivial. Several wiring architectures are conceivable, and one particular example is described in [12]. Nevertheless, we could show (see Appendix D) that LW PUFs can be broken by essentially the same methods as XOR arbiter PUFs, namely logistic regression with Rprop gradient descent. Our experiments also proved that not only the original wiring of [12] can be learned, but also random wirings or mappings between the external input and the internal challenges to the k arbiters. This suggests that LW PUFs in their current form are not secure as strong PUFs. They remain candidates for obfuscating PUFs, though.

Feed-Forward Arbiter PUF. Feed-Forward Arbiter PUFs (FF-Arb PUFs) have been introduced, too, to make ML attacks more difficult [5, 7]. Again, one of their disadvantages is their decreased stability; for example, the stability of a FF-Arb PUF with seven FF-loops decreases to 90.16 % if considered over a temperature variation of 45° C [5]. Despite the fact that perceptrons and SVM fail to learn these structures, more involved machine learning techniques are capable of attacking them. Some very recent ML-experiments conducted in our group suggest that FF-Arb PUFs are vulnerable to so-called evolutionary algorithms. These are particularly suited for our task, since they naturally allow us to “feed” or “include” our knowledge about the internal layout or wiring of the PUF in the fitness evaluation step of the ML algorithm. The CRP-data samples were again generated via a linear additive delay model. This means that even FF-Arb PUFs are no strong PUFs, but that they can mainly serve as obfuscating PUFs.

Coating PUF. Coating PUFs [6] are no strong PUFs according to Definition 3.2: Eve can query the oracle that she is provided with for all possible challenge-response-pairs of the coating PUF within very short time (there is essentially just one CRP, which gives the local capacitances determined by all sensors). Please note that without the oracle, Eve would find it hard to determine these capacitances: Any invasive attack would change the capacitance and destroy the responses. In other words, coating PUFs nicely illustrate why it is important to include the oracle access of Eve in Definition 3.2. Coating PUFs are no strong PUFs, but constitute the archetypical candidate of an obfuscating PUF.

¹Please note that the other obvious option to set up an XOR-based arbiter PUF, namely to apply independent challenges C_i^1, \dots, C_i^k at the k arbiters, does not make sense: In this case, we can fix all challenges apart from those challenges applied to one of the k arbiters. This creates a structure that behaves essentially like a single arbiter. Then, we can machine learn the behavior of this single arbiter by the known methods [5]. Subsequently, we can go on to fix other challenges and vary the single challenges applied to another arbiter, etc., until we have successively machine learned the behavior of the whole structure.

SRAM-based PUF/Butterfly PUF. By the very same argument as above, SRAM-based PUFs/Butterfly PUFs [10, 11] are no strong PUFs, but candidates for obfuscating PUFs.

Summary. The following table summarizes our findings. As noted earlier, we cannot positively prove that one of the listed PUFs actually is an obfuscating or strong PUF according to Definitions 3.2 and 4.1. They can merely have the status of candidates. On the other hand, one can disprove that a PUF fulfills Definitions 3.2 or 4.1, leading to “no”-entries in the table. The table shows that with one exception, all currently existing “PUFs” are either candidates for strong PUFs or obfuscating PUFs, but not for both, and that also every known PUF is a candidate for at least one of the two notions.

	Opt.	Int.Opt.	R.Osc.	Arb.	XOR	LW	FF	Coat.	SRAM
Obf. PUF	No	Cand.	Cand.	Cand.	Cand.	Cand.	Cand.	Cand.	Cand.
Str. PUF	Cand.	Cand.	No	No	No	No	No	No	No

5.1 Comparison with Previous Notions

For comparison, we also provide a classification of current PUF candidates with respect to the previous notions of Physical One-Way Functions (POWF, Def. A.2), Physical Random Function (PRF, Def. B.1), Strong Physical Unclonable Functions (Strong PUF as in Def. C.1) and Weak Physical Unclonable Functions (Weak PUF, Def. C.1). As the reader may verify relatively easily, the entries in the following table follow from the discussions in section 2 and 5, and from the new machine learning results presented in Appendix D.

Please note that it is not fully clear how Def. B.1 should be interpreted with respect to Coating PUFs and SRAM-based PUFs, whence we leave the respective classification open.

	Opt.	Int.Opt.	R.Osc.	Arb.	XOR	LW	FF	Coat.	SRAM
POWF	No	No	No	No	No	No	No	No	No
PRF	No	Cand.	No	No	No	No	No	??	??
Strong PUF	No	No	No	No	No	No	No	No	No
Weak PUF	No	No	No	No	No	No	No	Cand.	Cand.

The two last tables seem to support our proposed formalism of strong t -PUFs and obfuscating t -PUFs, and the split of PUFs into two different notions.

6 Differing Security and Practicality Features in Applications

We will now analyze the differing properties of strong PUFs and obfuscating PUFs in a concrete security application. We chose an identification-like scenario that was among the first applications ever suggested for PUFs [1, 2]. Strong PUFs and obfuscating PUFs have notable differences therein.

6.1 Setting

The situation we consider is as follows: Alice wants to identify herself towards a central server via a certain physical token that she holds. The token is put into a terminal, which communicates with the server. This basic setting applies in many practical situations, for example bank cards, branded products, identity cards, access cards, etc. [1, 2]. The application of PUFs to both scenarios has been described intensively in the literature.

6.2 Review of Basic PUF-Protocols for Identification and Labeling

Before we analyse the differences, we will quickly repeat the basic protocols and schemes for the convenience of the reader. Readers familiar with the protocols may skip to section 6.3.

If strong PUFs are applied to the above identification setting, the protocol works as follows [1]: The central server must know a list of CRPs. It sends randomly chosen challenges C_i from the list to the terminal, which returns the responses M_{C_i} obtained from the label. These are compared to the CRPs in the server's list. Any CRP cannot be used more than once in this setting, meaning that the server must have a very large list on stock. In order to reduce the storage requirements, protocols that allow refreshment of the CRPs at the central server are possible [8].

In the case of an obfuscating PUF S that contains a key K_S , two possibilities are conceivable. Option one is that the central server stores K_S , and that the central server executes a symmetric identification protocol on the basis of K_S with the labeled item. The communication between them is executed via the terminal.

Option two is that the labeled item stores a public key/private key pair (pk, sk) . The private key sk is encrypted with the obfuscated key K_S via a one time pad scheme, and the encrypted value $Enc(sk)$ is stored on the labeled item. The public key pk is stored in plain.

In order to authenticate (pk, sk) , the public key pk must be signed (or 'tagged') by some trusted authority CA via its key $SigKey_{TA}$. The terminal holds the corresponding public verification key $VerKey_{TA}$. The verification of a label by the terminal proceeds in two steps: (i) The labeled item passes on the CA 's 'tag' and the key pk to the terminal. (ii) The terminal verifies the tag via use of $VerKey_{TA}$. (iii) The terminal executes an asymmetric identification protocol with the labeled item on the basis of pk . During this protocol, the labeled item decrypts $Enc(sk)$, and uses sk to generate its responses. This setting allows even offline-verification of the label, i.e. without a connection to a central server.

6.3 Practicality Features

We will now illustrate the different practicality features of Strong and Obfuscating PUFs in the above example application.

Strong PUFs. Strong PUFs only allow secure labeling in connection with a central server, which must store a very large number of CRPs. The computational load on the labeled item is essentially zero, meaning that very cheap labels can be generated. Error correction on the measurement values can be carried out by the terminal.

Refreshment of CRPs is possible via a protocol given in [8], which reduces the storage requirements on the server. However, the protocol comes at the cost of reduced security properties (see section 6.4).

Obfuscating PUFs. Obfuscating PUFs allow verification of the label via a central server, too, but advantageously on the basis of one single short bitstring stored on the server. This string can be used multiple times and does not need to be refreshed. The label must execute a symmetric identification scheme, imposing some computational load on it. Alternatively, as described in section 6.1, even offline verification of the labels is possible. In this case, however, asymmetric schemes must be executed by the label itself, leading to a strong computational load on a mobile system.

6.4 Security Features

The different security features of Strong and Obfuscating PUFs in the above appliance are as follow.

Strong PUFs. Strong PUFs lead to secure labeling under the mere assumptions that the underlying PUF is secure. The scheme remains secure even if Eve has physical access to the labeled item and the PUF contained therein multiple times.

If the CRPs at the server are refreshed via the protocol of [8], one necessary assumption is again the security of the underlying strong PUF. The protocol brings about a number of additional assumptions, however: Firstly, unproven computational assumptions on the security of the cipher employed in the protocol (see [8]). Secondly, it can be shown that the protocol is not secure against an adversary who (i) eavesdrops the communication between the item, the terminal and the server, and who (ii) gains physical access to the PUF more than once. This enforces the slightly unrealistic assumption that Eve must not have access to the PUF more than once.

Obfuscating PUFs. Obfuscating PUFs allow secure labeling only under a number of additional assumptions, which go beyond the mere security of an obfuscating PUF. Let us start with the case of a central sever which stores K_S : We firstly must assume that the underlying MAC is computationally secure. Next, we face two hardware assumptions *on the labeled item*: We must suppose that the transfer from the key K_S to the unit where the MAC is computed on the item can be carried out securely, without allowing Eve invasive, semi-invasive or side channel attacks. Finally, we must assume that the MAC is not only computationally secure, but that it is also implemented securely in hardware on the item, and that no power analysis, emanation analysis or other side channel attacks are possible. This host of assumptions arises because obfuscating PUFs eventually are nothing else than a secure form of secret key storage. Their practical use has to be complemented by classical mathematical cryptoschemes, which bring about the standard, well-known attacks.

The security analysis in the case of offline labeling, where we do not use a central server, but employ a central tagging authority and a public key/private key pair stored on the item, is similar.

6.5 Summary of Practicality and Security

Strong PUFs can lead to very nice security properties under very few security assumptions. Obfuscating PUFs, to the contrary, require more security assumptions, and can lead to a significant computational load on mobile systems. On the good side, they sometimes offer practicality advantages, such as offline label verification in our above example. Our discussion reassures the differences of strong and obfuscating PUFs also on the application side.

7 General Emulation: Strong from Obfuscating?

Let us discuss one final aspect in the relation between strong and obfuscating PUFs. It seems at first glance that a strong t -PUF S' can be constructed from an obfuscating t -PUF S in the following way: S' includes S as a subsystem. Whenever S' is presented with a challenge C , it derives the key K_S from S . Then, it computes and outputs $PRNG(C, K_S)$, for some suitably chosen and implemented pseudo-random number generator $PRNG$.

The construction seems to bring about a general simulation theorem, but nevertheless, there are a few problems associated it. First of all, it requires a number of additional assumptions. They have already been touched on in section 6.4:

1. The PRNG must be computationally secure (as a mathematical function).
2. The PRNG must be implemented physically in a secure manner. In particular, no side channel attacks must be possible (power consumption, emanation, or timing analysis, etc).

3. The on-circuit transport of K_S from the obfuscating PUF to the PRNG must be handled securely.
4. The underlying obfuscating PUF must be secure according to Def. 4.1.

Furthermore, from a formal perspective, the construction does not fulfill the manufacturer resistance condition in item 1 of Def. 3.2. The manufacturer could read out K_S , copy it and use it in several systems, which would show indistinguishable behavior.

Finally, the above construction requires that the PUF can handle the computational load of the PRNG. This practicality restriction, together with the above list of additional assumptions and the mentioned definitional issues, makes it preferable in any case to directly construct strong PUFs, even though our construction shows that they can to some extent be derived from obfuscating PUFs.

8 Summary and Conclusions

Our topics in this paper have been as follows:

- (i) We discussed some logical and conceptual issues in the existing definitions of PUFs.
- (ii) We proposed to split the current notion of a PUF into two new notions, strong PUFs and obfuscating PUFs. Strong PUFs try to express the original concept of a POWF/PRF in a consistent manner [2, 3], while the purpose of obfuscating PUFs is to pinpoint those properties relevant for key obfuscation [3].
- (iii) We suggested a new framework for these two notions, and presented corresponding formal definitions.
- (iv) We provided classifications of existing PUFs with respect to our new framework and with respect to previous definitions.
- (v) In this classification, we used some new machine learning results of our group. For the first time, these results show that XOR-Arbiter and Feed-Forward Arbiter PUFs can be attacked on large scales.
- (vi) We illustrated important security and practicality differences between strong PUFs and obfuscating PUFs in practical applications.
- (vii) We discussed an emulation strategy by which strong PUFs can (or cannot) be constructed from obfuscating PUFs.

Our findings also point to the fact that even though strong PUFs are a powerful and elegant security tool, currently no integrated electrical implementations exist. The investigation of such structures seems to be a rewarding future research goal.

Acknowledgements

The authors would like to thank Qingqing Chen, György Csaba, Jon Finley, Stefan Katzenbeisser, Heike Busch, Paolo Lugli, Christian Osendorfer, Jürgen Schmidhuber, Ulf Schlichtmann, Vera Stoyanova, Martin Stutzmann, and Ju Xueming for enjoyable and helpful discussions.

References

- [1] R. Pappu, B. Recht, J. Taylor, N. Gershenfeld, *Physical One-Way Functions*, Science, vol. 297, pp. 2026-2030, 20 September 2002.
- [2] R. Pappu, *Physical One-Way Functions*, PhD Thesis, MIT, 2001.
- [3] Blaise Gassend, *Physical Random Functions*, MSc Thesis, MIT, 2003.
- [4] Blaise Gassend, Daihyun Lim, Dwaine Clarke, Marten van Dijk, Srinivas Devadas: *Identification and authentication of integrated circuits*. Concurrency and Computation: Practice & Experience, pp. 1077 - 1098, Volume 16, Issue 11, September 2004.
- [5] Daihyun Lim: *Extracting Secret Keys from Integrated Circuits*. MSc Thesis, MIT, 2004.
- [6] Pim Tuyls, Geert Jan Schrijen, Boris Skoric, Jan van Geloven, Nynke Verhaegh, Rob Wolters *Read-Proof Hardware from Protective Coatings*. CHES 2006: 369-383
- [7] G. Edward Suh, Srinivas Devadas: *Physical Unclonable Functions for Device Authentication and Secret Key Generation*. DAC 2007: 9-14
- [8] P. Tuyls, B. Skoric. *Strong Authentication with PUFs*. In: Security, Privacy and Trust in Modern Data Management, M. Petkovic, W. Jonker (Eds.), Springer, 2007.
- [9] P. Tuyls, B. Skoric, T. Kevenaar (Eds.): *Security with Noisy Data*. Springer 2007.
- [10] Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, Pim Tuyls: *FPGA Intrinsic PUFs and Their Use for IP Protection*. CHES 2007: 63-80
- [11] Sandeep S. Kumar, Jorge Guajardo, R. Maes, Geert Jan Schrijen, Pim Tuyls: *The Butterfly PUF: Protecting IP on every FPGA*. HOST 2008: 67-70.
- [12] Mehrdad Majzoobi, Farinaz Koushanfar, Miodrag Potkonjak: *Lightweight secure PUFs*. ICCAD 2008: 670-673.
- [13] T. Bäck: *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, USA, 1996.
- [14] H.P. Schwefel: *Evolution and optimum seeking*. Wiley, New York, 1995.
- [15] I. Rechenberg: *Evolutionstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Fromman-Holzboog, Stuttgart, Germany, 1973.
- [16] H.-P. Schwefel: *Numerical Optimization of Computer Models*. John Wiley and Sons, LTD, 1981.
- [17] O. Goldreich, *The Foundations of Cryptography*. Volume 1 & 2, ISBNs: 0-521-79172-3 (Vol. 1)& 0-521-83084-2 (Vol. 2), Cambridge University Press, 2001/2004.
- [18] M. Riedmiller, H. Braun: *RPROP-A fast adaptive learning algorithm*. Proc. of ISCIS VII, Universitat, 1992
- [19] C.M. Bishop: *Patern recognition and machine learning*. Springer New York, 2006
- [20] Jakob D. Bekenstein: *How does the Entropy/Information Bound Work?* Foundations of Physics, vol. 35, issue 11, pp. 1805-1823. Latest version also from <http://arxiv.org/abs/quant-ph/0404042>.

A Physical One-Way Functions

The notion of a ‘‘Physical One-Way Function’’ has been defined by R. Pappu in [2]. It definitely owns the merit of being the first formalization attempt in the field. Before giving his definition, Pappu introduces some notation, which was developed with hindsight to his optical PUF [1, 2].

Notation A.1 (Notation for Physical One-Way Functions, as in [2]). *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 .

On the basis of this notation, [2] devises the following definition.

Definition A.2 (Physical One-Way Functions, as in [2]). *$f : \{0, 1\}^l \times \{0, 1\}^k \rightarrow \{0, 1\}^n$ 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 $p(\cdot)$ 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 $O(\text{poly}(l))$ 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.

B Physical Random Functions

Gassend [3] provides another definition, which does not emphasize the non-invertability of PUFs, but their unpredictability or ‘‘randomness’’.

Definition B.1 (Physical Random Functions, quoted from [4]). *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.

In the above definition, the terms short and polynomial are relative to the size of the device.

C Physical Unclonable Functions

Finally, Tuyls et al. give a third specification of PUFs [10]. It is not marked as a definition in their text, whence we quote it as a “description” here.

Description C.1 (Physical Unclonable Functions, quoted from [10]). *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) , $i = 1, \dots, N$, 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.

D Machine Learning Attacks on Physical Unclonable Functions

It has long been known that PUFs can potentially be attacked via machine learning (ML) techniques. In these attacks, an adversary collects many CRPs from the PUF and uses them to train an ML algorithm. If successful, the trained algorithm can subsequently imitate the PUF, thereby breaking its security. For example, it has been shown relatively early that perceptrons [4] and Support Vector Machines (SVMs) [5] suffice to break standard arbiter PUFs.

In order to avoid these attacks, strengthened versions of the arbiter PUF have been suggested: XOR arbiter PUFs [5, 7], lightweight PUFs [12] and feed-forward arbiter PUFs [5, 7]. Nevertheless, we were able to successfully break all these variants by suitable ML techniques.

Logistic Regression, XOR Arbiters and Lightweight PUFs. Logistic regression is a well established, generic ML method [19]. As our recent efforts show, it can be applied successfully to the cryptanalysis of the XOR arbiter and the lightweight PUF.

As a starting point, we used the standard linear model of the arbiter PUF described in [5]. In this model, it is assumed that the challenge \vec{C} of a b -stage arbiter PUF is a string from the set $\{-1, 1\}^b$ (instead of $\{0, 1\}^b$). Correspondingly, the arbiter output is encoded into a binary response $t \in \{-1, 1\}$ (instead of $\{0, 1\}$). Under these provisions, the final output t can – in dependence of the final propagation delay difference Δ at the end of the sequence of multiplexers – be written as

$$t = \text{sgn}(\Delta) = \text{sgn}(\vec{w}^T \vec{P}), \quad (2)$$

$$\text{where } P^i = \prod_{j=i}^b C^j, \quad \vec{P} = (P^1, P^2, \dots, P^n, 1)^T, \quad (3)$$

and where the parameter vector \vec{w} is a direct function of the (unknown) inner subdelays of the PUF. For the details, see [5].

In order to apply the above model to an XOR-arbiter structure, in which the final output t_{xor} is computed as the parity of k single arbiter outputs, one writes:

$$t_{xor} = \text{sgn}\left(\prod_{r=1}^k t_r\right) = \text{sgn}\left(\prod_{r=1}^k \Delta_r\right) = \text{sgn}\left(\prod_{r=1}^k \vec{w}_r^T \vec{P}_r\right). \quad (4)$$

From the above formalism, we can derive an obvious ML decision boundary by

$$\prod_{r=1}^k \vec{w}_r^T \vec{P}_r = 0. \quad (5)$$

Logistic regression now progresses by optimizing the vectors \vec{w}_r in (5) such that the likelihood of observing the CRPs of a training set \mathcal{M}

$$p(\mathcal{M}|\vec{w}) = \prod_{m \in \mathcal{M}} \sigma(t^m \cdot \prod_{r=1}^k \vec{w}_r^T \vec{P}_r^m) \quad (6)$$

becomes maximal. There are various methods to carry out this optimization. Our experiments showed that Rprop gradient descent [18] performs best, and that standard gradient descent and also iterative reweighted least squares do not work for the considered problem.

This leads to a prediction rate on test sets of 99% for arbiter PUFs, provided that the vectors \vec{w}_r have been well trained on sufficiently large training sets. Empirically, a first estimate for the required number of CRPs is $50 \cdot k \cdot (b + 1)$ or $O(kb)$ (as above, k denotes the number of arbiters and b the arbiter length). The vectors \vec{w}_r are considered as well trained if the corresponding decision boundary (5) correctly classifies over 99% of the training set.

However, the algorithm does not yield well trained vectors for every initial choice of the \vec{w}_r . Therefore it has to be evaluated repeatedly on the training set, with randomly initialized parameter vectors \vec{w}_r , until an adequate solution is found. By this approach the security of the XOR Arbiter PUF and/or the Lightweight PUF with different internal wirings can be broken, as shown below in Table 1. All described PUFs are of type 4-XOR, 64 bit.

	equal challenges	random challenges	lightweight challenges
training set size [CRPs]	19000	19000	49000
mean computing time [min]	0.7	58.5	12
prediction rate on test set	99.1%	98.9%	98.7%

Table 1: Performance of the logistic regression approach for different XOR and Lightweight PUFs. The examined PUFs are all XOR-arbiter type PUFs, but differ by the precise mapping of the external challenge to the internal challenges applied at the individual internal arbiter PUFs. In the case of equal challenges, the challenge for all the individual internal arbiter PUFs is the same as the external challenge. For random challenges, the overall, external challenge is (pseudo) randomly transformed to individual challenges of the 4 internal arbiters. The mapping of the lightweight PUF is described in [12]. The mean computing time was determined on a standard quad core PC.

Evolution Strategies and Feed-Forward Arbiters. Evolution Strategies (ES) [14] belong to a class of ML techniques called Evolutionary Algorithms. They are inspired by the biological evolution of a population of individuals under certain environmental conditions. The population repeatedly undergoes the evolutionary steps of evaluation, environment selection, partner selection, recombination and mutation, until individuals are found which match some previously fixed environmental conditions very well.

In our case, an individual is given by a concrete instantiation of the runtime delays in a PUF (or by the vector \vec{w} from equation (5)). The environmental fitness is determined by how well this individual (re-)produces the correct CRPs of the target PUF as output. The outputs of the individual are computed by a linear additive delay model from its subdelays (or from \vec{w}), and are compared to several known outputs of the target PUF structure. The best individuals are selected. In the following recombination step, the remaining individuals mutually exchange part of their 'genome'/their individual subdelays, in order to form descendants. In the final mutation step, some of the subdelays are varied randomly, and the process starts anew, producing more and more 'generations' of the population. Evolutionary algorithms bear the advantage that our full knowledge of the PUF architecture can be exploited by building a corresponding model into the fitness evaluation step.

We used a standard implementation of ES with the ES standard meta-parameters [13]: Population size of (6,36), comma-best-selection, and a global mutation operator of $\tau = \frac{1}{\sqrt{(n)}}$. As the training data underlying the experiments, we used a set of 50.000 CRPs with random subsets of 2.000 CRPs for the evaluation step of the individuals. These CRPs were generated on the basis of a linear additive delay model. The subdelays in the stages were drawn according to a uniform distribution with parameters motivated by the standard fabrication delays that may occur in such a structure. The architecture of the examined structures is shown in Figure 3.

Our ML results are depicted in Figure 4. For each FF-Arb PUF, it shows the best out of 10 runs that were conducted on the structure. The x -axis displays the number of generations, the y axis the prediction error. As shown, in all but one case we have been able to successfully learn the PUFs with rates better than 95 % in only 600 generations. Since ES are a probabilistic method, additional runs can be expected to yet further improve the prediction accuracy. Also other meta-parameters like bigger population sizes, more generations or smaller τ can optimize the performance further. Please note that our accuracy is significantly better than the stability of a FF-arbiter with 7 FF-loops under a temperature variation of 45°C, which is only 90.16 % [5]. In this sense, our experiments indicate that the FF arbiter can be fully broken by ES.

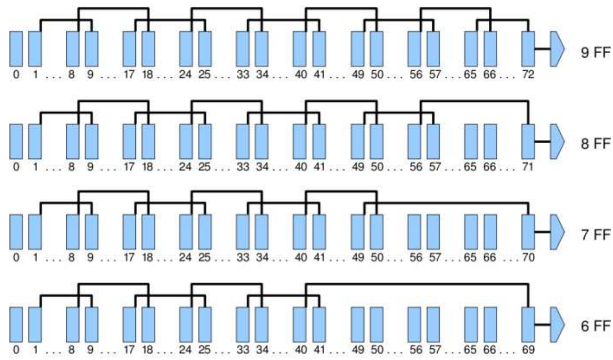


Figure 3: The feed-forward architectures we used in our learning experiments.

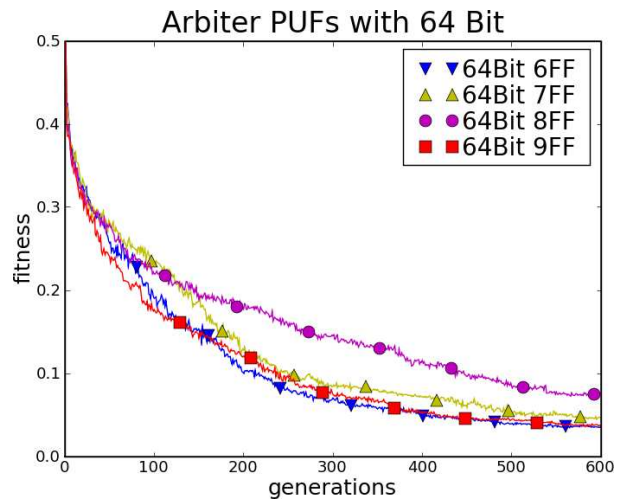


Figure 4: The best of 10 runs on each of the architectures.