Converse PUF-Based Authentication

Ünal Kocabaş[1], Andreas Peter[1],
Stefan Katzenbeisser[1], and Ahmad-Reza Sadeghi[2]

[1] Technische Universität Darmstadt (CASED), Germany
[2] Technische Universität Darmstadt & Fraunhofer SIT Darmstadt, Germany
{unal.kocabas,ahmad.sadeghi}@trust.cased.de, andreas.peter@cantab.net,
skatzenbeisser@acm.org

Abstract. Physically Unclonable Functions (PUFs) are key tools in the construction of lightweight authentication and key exchange protocols. So far, all existing PUF-based authentication protocols follow the same paradigm: A resource-constrained prover, holding a PUF, wants to authenticate to a resource-rich verifier, who has access to a database of pre-measured PUF challenge-response pairs (CRPs). In this paper we consider application scenarios where all previous PUF-based authentication schemes fail to work: The verifier is resource-constrained (and holds a PUF), while the prover is resource-rich (and holds a CRP-database). We construct the first and efficient PUF-based authentication protocol for this setting, which we call *converse* PUF-based authentication. We provide an extensive security analysis against passive adversaries, show that a minor modification also allows for authenticated key exchange and propose a concrete instantiation using controlled Arbiter PUFs.

Keywords: Physically Unclonable Functions (PUFs), Authentication, Key Exchange.

1 Introduction

With rapid improvements in communication technologies, networks have become widespread, connecting both low-cost devices and high-end systems. Low-cost devices, such as RFID-tags, sensor nodes, and smart cards are likely to form the next generation pervasive and ubiquitous networks. Such networks are designed to store sensitive information and transmit this information to participants over a potentially insecure communication channel. Due to the potentially sensitive data they handle, security features such as authentication and encrypted data transfer and required. At the same time, the deployed security features must be extremely lightweight to fit the application scenario.

Physically Unclonable Functions [12], security primitives that extract noisy secrets from physical characteristics of integrated circuits (ICs), have emerged as trust anchors for lightweight embedded devices. Instead of relying on heavyweight public-key primitives or secure storage for secret symmetric keys, PUFs can directly be integrated in cryptographic protocols. PUFs have successfully

S. Katzenbeisser et al. (Eds.): TRUST 2012, LNCS 7344, pp. 142–158, 2012.

been used in the context of anti-counterfeiting solutions that prevent cloning of products, and in the construction of various cryptographic protocols, involving identification and authentication.

In this paper we are merely concerned with PUF-based authentication protocols. All previous approaches, including [24,15,9], considered the problem of authenticating a lightweight device (called prover) containing a PUF to a remote entity (called verifier), which has more storage and processing capabilities. In particular, the verifier is required to store a database of measured PUF *challenge-response pairs* (CRPs). In order to perform the authentication, the verifier sends a random challenge to the prover, who has to measure the PUF on the challenge and respond with the measured PUF response. If the obtained response matches the one stored in the CRP, the prover is authenticated. Note that CRPs cannot be re-used since this would enable an adversary to mount replay attacks; furthermore, it would allow tracing of the tag. Besides this issue, some PUFs are subject to model-building attacks [25], which allow to obtain a model of the PUF in use by observing the PUF challenge-response pairs contained in the protocol messages.

In this work we consider PUF-based authentication protocols tailored towards a different scenario in which the verifier $\mathcal{V}$ is a very resource-constrained (yet PUF-enabled) device, while the prover $\mathcal{P}$ has comparably rich computational resources. For example, one can consider the scenario in which a sensor node (acting as verifier) wants to authenticate a sink (prover) in order to transmit sensitive sensor readings. In this setting, all currently available PUF-based authentication protocols are not applicable, since the roles of prover and verifier are reversed (simply swapping the roles of verifier and prover in traditional protocols does not work either, since a resource-constrained device is not able to keep a CRP database). In this paper we therefore propose a novel PUF-based authentication protocol that works in this situation: The prover $\mathcal{P}$ holds a CRP-database, while the lightweight verifier $\mathcal{V}$ has access to the PUF. Due to this converse approach of using the PUF in authentication, we call protocols that follow this new paradigm *converse PUF-based authentication protocols*. As a second feature of our protocol, which is in contrast to all previous approaches, our construction *never* needs to transmit PUF responses (or hashes thereof) over the channel, which effectively prevents passive model-building attacks as well as replay attacks. Since in this work, we deal with passive adversaries only, we see our solution as the first step in this converse approach and hope to see more work on this matter in the future.

1.1 Contributions

In summary, the paper makes the following contributions:

Introduction of a New Paradigm for PUF-Based Authentication. We introduce the paradigm of converse PUF-based authentication: In this setting a prover $\mathcal{P}$ holds a CRP-database, while a lightweight verifier $\mathcal{V}$ has access to a PUF.

First Construction. Based on an idea introduced in [5], we construct the first converse PUF-based authentication protocol, which is at the same time very efficient. It uses a controlled PUF at the verifier and a CRP database at the prover. A key feature is that during the protocol only a random tag and two PUF-challenges are exchanged over the communication channel; this effectively prevents model building attacks.

Security Analysis. We provide an extensive security analysis of the new protocol and show that it is secure against passive adversaries. We deduce precise formulae that upper bound the success probability of a worst-case adversary after having seen a certain number of protocol transcripts.

Authenticated Key Exchange. Finally, we show that a minor modification of our authentication protocol allows the two participants to agree on a common secret key. This basically comes for free, since this modification only amounts to the evaluation of one additional hash function at both sides.

1.2 Outline

After presenting a brief summary of PUFs and their properties, fuzzy extractors, and controlled PUFs in Section 2, we introduce our converse PUF-based authentication protocol including a proof of correctness in Section 3. Then, in Section 4 we discuss the security model we consider and prove our protocol secure against passive adversaries. Finally, implementation details are given in Section 5. We conclude with a summary and some possible directions for future work in Section 6.

2 Background and Related Work

PUFs exploit physical characteristics of a device, which are easy to measure but hard to characterize, model or reproduce. Typically, a stimulus, called challenge C, is applied to a PUF, which reacts with a response R. The response depends on both the challenge and the unique intrinsic randomness contained in the device. A challenge and its corresponding response are called a *challenge-response pair* (CRP). Typical security assumptions on PUFs include [21]:

- *Unpredictability:* An adversary A cannot predict the response to a specific PUF challenge without modeling its intrinsic properties. Moreover, the response R_i of one CRP (C_i,R_i) gives only a small amount of information on the response R_j of another CRP (C_j,R_j) with $i \neq j$.
- *Unclonability:* An adversary A cannot emulate the behavior of a PUF on another device or in software, since the behavior is fully dependent on the physical properties of the original device.
- *Robustness:* The outputs of a PUF are stable over time; thus, when queried with the same challenge several times, the corresponding responses are similar (which opens the possibility to apply an error correcting code in order to obtain a stable response).

PUFs meeting these assumptions provide secure, robust and low cost mechanisms for device identification and authentication [24,30,23,26], hardware-software binding [13,16,14,7] or secure storage of cryptographic secrets [8,33,18,4]. Furthermore, they can be directly integrated into cryptographic algorithms [1] and remote attestation protocols [27].

Among different PUF architectures, we focus on electronic PUFs, which can be easily integrated into ICs. They essentially come in three flavors: *Delay-based PUFs* are based on digital race conditions or frequency variations and include arbiter PUFs [17,23,19] and ring oscillator PUFs [12,29,22]. *Memory-based PUFs* exploit the instability of volatile memory cells after power-up, like SRAM cells [13,15], flip-flops [20,32] and latches [28,16]. Finally, *Coating PUFs* [31] use capacitances of a special dielectric coating applied to the chip housing the PUF.

Arbiter PUFs. In this paper we use *Arbiter PUFs* (APUF) [17], which consist of two logical paths, controlled by a challenge. Both paths get triggered at the same time. Due to the inherently different propagation delays induced by manufacturing variations, one of the two paths will deliver the signal faster than the other; a digital arbiter finally determines which of the two signals was faster and produces a one-bit response. The number of challenge-response pairs is typically exponentially large in the dimensions of the APUF, which makes them a good candidate to be used in authentication mechanisms.

However, it was claimed in [25] that APUFs are subject to model building attacks that allow predicting responses with non-negligible probability, once an attacker has full physical access to the APUF or can record sufficiently many challenge-response pairs. Further, the response of an APUF cannot be used directly as a cryptographic key in an authentication mechanism without post-processing, since two queries of the same challenge may give slightly different responses due to noise. In order to counter these problems, additional primitives must be used: *Fuzzy Extractors* (FE) [6] and *Controlled PUFs* [9].

Fuzzy Extractors. The standard approach to make the PUF responses stable, is to use Fuzzy Extractors [6] consisting of a *setup phase*, an *enrolment phase* and a *reconstruction phase*.

In the setup phase, an error-correcting binary[1] linear $[\mu, k, d]$-code $\mathcal{C}$ of bit length μ, cardinality 2^k, and minimum distance d is chosen. Due to the choice of parameters, the code can correct up to $\lfloor \frac{d-1}{2} \rfloor$ errors. There are many known ways to construct such codes for given parameters [6], and we just mention here that we need to set the parameter μ to be the bit length of the output of the used PUF (some care has to be taken when choosing the amount of errors the code needs to correct, see [3]).

In the enrolment phase, denoted by $\mathsf{FE.Gen}$, which is carried out before the deployment of the chip in a device in a trusted environment, for any given PUF response R, we choose a random codeword $\gamma \xleftarrow{U} \mathcal{C}$ and compute the *helper data* $h := \gamma \oplus R$. Later, during the reconstruction phase (denoted by $\mathsf{FE.Rep}$), for

[1] We restrict our attention to binary codes (i.e., codes over the binary Galois field $\mathbb{F}_2$), although the same discussion can be done for non-binary codes as well.

any given PUF response R and corresponding helper data h, we first compute $W := R \oplus h$, and then use the decoding algorithm of the error correcting code $\mathcal{C}$ on W, which outputs the same codeword γ that we randomly picked in the enrolment phase.

Controlled PUFs. If one requires a uniformly distributed output (which a PUF usually does not provide), one can apply a cryptographic hash function $H : \{0,1\}^* \longrightarrow \{0,1\}^n$ to the output γ of the FE [3]. Here, we will always treat such a hash function H as a random oracle [2] which ensures that the output is uniformly distributed in $\{0,1\}^n$. Usually, an LFSR-based Toeplitz Hash function is used in order to implement this *privacy amplification phase* because of its low cost. The resulting combined primitive, i.e., applying the hash function H to the output of the FE, which itself was applied to the PUF, is called a *controlled PUF*.

3 Converse PUF-Based Authentication

All currently existing PUF-based (unilateral, two-party) authentication protocols (e.g., [24,15,9]) follow the same paradigm: A prover $\mathcal{P}$, who has access to a PUF, wants to authenticate himself to a verifier $\mathcal{V}$ who holds a database of challenge-response pairs (CRP) of $\mathcal{P}$'s PUF. In this section, we propose a new PUF-based authentication protocol that actually works the other way around: The prover $\mathcal{P}$ holds a (modified and reduced) CRP-database, while the verifier $\mathcal{V}$ has access to a PUF. Due to this converse approach of using the PUF in the authentication, we call protocols that follow this new paradigm *Converse PUF-based Authentication Protocols*.

3.1 Protocol Description

We consider a controlled PUF consisting of an underlying physical PUF (denoted by **PUF**), the two procedures $\mathsf{FE.Gen}$ and $\mathsf{FE.Rep}$ of the underlying Fuzzy Extractor (FE), and a cryptographic hash function $H : \{0,1\}^* \longrightarrow \{0,1\}^n$.

Now, as in usual PUF-based authentication, our protocol needs to run an *enrolment phase* in order to create a CRP-database on the prover's side. We note that this database will not consist of the actual CRPs of **PUF** but of responses of the controlled PUF (i.e., the PUF challenges C, some helper data h and hash values $H(\gamma)$ for FE outputs γ). More precisely, in the enrolment phase the prover $\mathcal{P}$ sends some random PUF challenge C to the verifier $\mathcal{V}$, who runs the enrolment phase of the FE on $\mathbf{PUF}(C)$, which outputs a value γ and some helper data h. Then, $\mathcal{V}$ returns the values $R(C,h) = H(\gamma)$ and h to $\mathcal{P}$. The prover $\mathcal{P}$ stores this data together with the PUF challenge in a database $\mathcal{D}$. These steps are repeated ρ times in order to generate a database $\mathcal{D}$ of size ρ. The described procedure is summarised in Fig. 1.

Now, whenever $\mathcal{P}$ needs to authenticate himself to $\mathcal{V}$, the following *authentication phase* is run: First $\mathcal{V}$ sends a random $0^n \neq \Delta \xleftarrow{U} \{0,1\}^n$ to $\mathcal{P}$. Then, $\mathcal{P}$

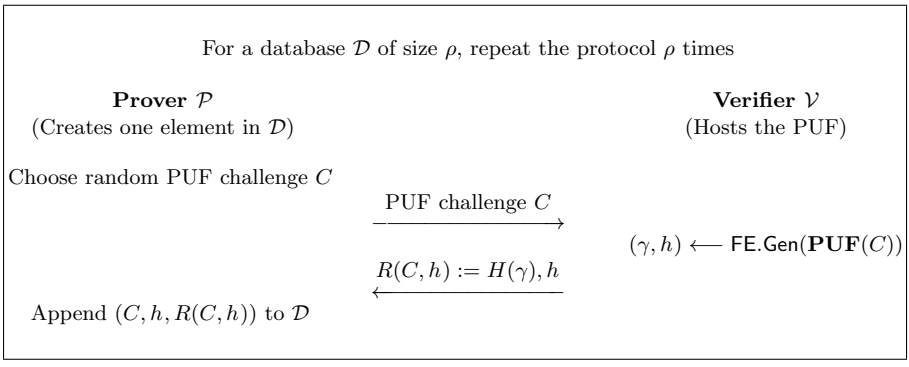


Fig. 1. Enrolment phase: Creating $\mathcal{P}$'s database $\mathcal{D}$

searches through his database $\mathcal{D}$ in order to find two elements $(C_1, h_1, R(C_1, h_1))$ and $(C_2, h_2, R(C_2, h_2))$ such that $\Delta = R(C_1, h_1) \oplus R(C_2, h_2)$, and sends the pairs (C_1, h_1) and (C_2, h_2) to $\mathcal{V}$. In other words, he is looking for two controlled PUF outputs whose XOR is Δ. If no such elements exist in $\mathcal{D}$, $\mathcal{P}$ just sends $(C_1, h_1) = (C_2, h_2)$ to $\mathcal{V}$, where both the PUF challenge C_1 and the helper data h_1 are chosen at random. In this case the authentication fails; we will choose the protocol parameters in a way that this happens only with small probability. Now, $\mathcal{V}$ uses the reconstruction phase of the FE twice – once on input $\mathbf{PUF}(C_1)$ and h_1, and once on input $\mathbf{PUF}(C_2)$ and h_2 which output two code words γ_1 and γ_2, respectively. After applying the hash function H to this (yielding values $R(C_1, h_1) = H(\gamma_1)$ and $R(C_2, h_2) = H(\gamma_2)$, respectively), $\mathcal{V}$ checks whether $R(C_1, h_1) \oplus R(C_2, h_2) = \Delta$. If equality holds, $\mathcal{V}$ sends the message $M = \top$ back to $\mathcal{P}$ in order to indicate that $\mathcal{P}$ successfully authenticated himself to $\mathcal{V}$; else it returns $M = \bot$, signaling that the authentication failed. In a subsequent step, the responses may optionally be used to exchange a shared secret key (see Section 3.3). The complete authentication phase is summarised in Fig. 2.

3.2 Correctness of the Protocol

We recall that in the enrolment phase, the prover $\mathcal{P}$ gets a database $\mathcal{D}$ of size ρ containing pairs of *PUF-challenges* C, *helper data* h and corresponding *responses* $R(C, h)$ from the verifier $\mathcal{V}$. Furthermore, we recall that after applying the Fuzzy Extractor, we input the resulting output into a cryptographic hash function H. So if we require the FE's outputs to have $\kappa \geq n$ bits of entropy, we can think of the responses $R(C, h)$ as bitstrings taken *uniformly at random* from the set $\{0,1\}^n$ (cf. the Random Oracle Paradigm [2]). Here, we bear in mind that κ and n are public parameters of our authentication protocol that are being fixed in some setup phase.

In this section, we consider how the probability of a successful authentication of $\mathcal{P}$ is affected by the size ρ of $\mathcal{P}$'s database $\mathcal{D}$. In other words, we will give a

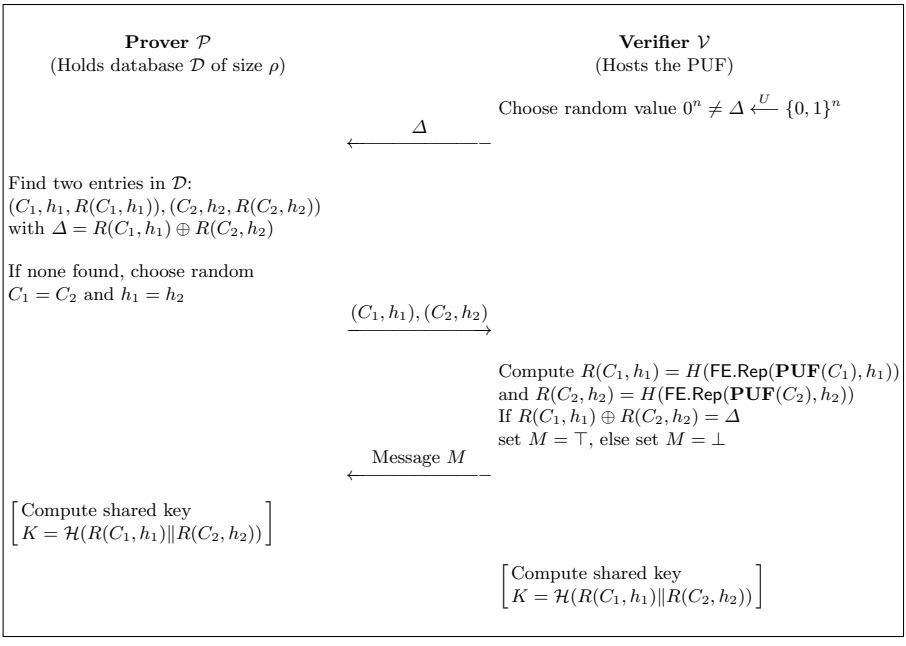


Fig. 2. Authentication phase: $\mathcal{P}$ authenticates himself to $\mathcal{V}$. As an optional step, both participants can compute a shared key K after the authentication.

lower bound on the size ρ of the database $\mathcal{D}$ in order for an authentication to be successful with a prescribed probability (assuming that both participants $\mathcal{P}$ and $\mathcal{V}$ honestly perform each step of the protocol). Here, *successful authentication* means that given a random $0^n \neq \Delta \xleftarrow{U} \{0,1\}^n$ there exist

$$(C_1, h_1, R(C_1, h_1)), (C_2, h_2, R(C_2, h_2)) \in \mathcal{D}$$

in $\mathcal{P}$'s database such that $R(C_1, h_1) \oplus R(C_2, h_2) = \Delta$.

Theorem 1. *If ρ denotes the size of $\mathcal{P}$'s database $\mathcal{D}$, then the probability of a successful authentication is*

$$\mathsf{Succ}^{\mathrm{Auth}}_{\mathcal{P},n}(\rho) := 1 - \left(1 - \frac{2}{2^n - 1}\right)^{\frac{\rho^2 - \rho}{2}}.$$

Proof. First of all, it is easy to see that only the responses $R(C, h)$ that are stored in the database $\mathcal{D}$ have an influence on the probability of a successful authentication, and so we think of $\mathcal{D}$ containing only responses $R = R(C, h)$ and forget about the PUF-challenges C and helper data h. Now, since the ρ different values $R(C, h)$ in $\mathcal{D}$ and the value Δ are uniformly distributed and independent in $\{0,1\}^n$, the probability of having a successful authentication amounts to the following:

For a set M, let $\binom{M}{2}$ denote the set of all subsets of cardinality 2 of M, whereas we denote elements of this set by pairs (R_1, R_2); so basically, this set consists of all *unordered* pairs (R_1, R_2), excluding self-pairs (R_1, R_1). Here, we consider the set $\binom{\{0,1\}^n}{2}$ which has precisely $\binom{2^n}{2}$ many elements. For the authentication, we are only interested in the XOR of two values in $\mathcal{D}$, so we want to look at the set $\binom{\mathcal{D}}{2}$ which has exactly $\binom{\rho}{2}$ many elements taken uniformly at random from $\binom{\{0,1\}^n}{2}$. We denote the set of all XOR's of any two elements in $\mathcal{D}$ by $\mathcal{D}^{\oplus}$, i.e., $\mathcal{D}^{\oplus} = \{R_1 \oplus R_2 \mid (R_1, R_2) \in \binom{\mathcal{D}}{2}\}$. Therefore, the probability of a successful authentication is the probability that $\Delta \in \mathcal{D}^{\oplus}$. Summing up, we have:

1. $\Delta \xleftarrow{U} \{0,1\}^n$ is sampled uniformly at random.[2]
2. The prover $\mathcal{P}$ has a database $\binom{\mathcal{D}}{2}$ of $\binom{\rho}{2}$ many elements taken uniformly at random from $\binom{\{0,1\}^n}{2}$.
3. For a random $(R_1, R_2) \xleftarrow{U} \binom{\{0,1\}^n}{2}$, the probability that we hit on Δ when XOR-ing R_1 and R_2 is $q := \frac{2^n}{\binom{2^n}{2}} = \frac{2}{2^n-1}$.
4. We are interested in the probability of a successful authentication, i.e., in the probability $\mathsf{Succ}^{\mathrm{Auth}}_{\mathcal{P},n}(\rho) = \Pr[\Delta \in \mathcal{D}^{\oplus}]$, where $\mathcal{D}^{\oplus} = \{R_1 \oplus R_2 \mid (R_1, R_2) \in \binom{\mathcal{D}}{2}\}$ and the latter probability is taken over all random $\Delta \xleftarrow{U} \{0,1\}^n$ and random $\binom{\mathcal{D}}{2} \subset \binom{\{0,1\}^n}{2}$.

In other words, we sample $\binom{\rho}{2}$ many times from $\binom{\{0,1\}^n}{2}$ with probability $q = \frac{2}{2^n-1}$ of success (i.e., hitting on Δ) on each trial, and ask for the probability of having *at least* $s = 1$ successes (i.e., hits on Δ). The probability of having exactly s successes is given by the *binomial probability formula*:

$$\Pr\left[s \text{ successes in } \binom{\rho}{2} \text{ trials}\right] = \binom{\binom{\rho}{2}}{s} q^s (1-q)^{\binom{\rho}{2}-s}.$$

Therefore, the probability of having $s = 0$ successes is $(1-q)^{\frac{\rho^2-\rho}{2}}$. Finally, this gives us the probability of having *at least* $s = 1$ successes, i.e., a successful authentication:

$$\mathsf{Succ}^{\mathrm{Auth}}_{\mathcal{P},n}(\rho) = 1 - \Pr\left[0 \text{ successes in } \binom{\rho}{2} \text{ trials}\right] = 1 - \left(1 - \frac{2}{2^n - 1}\right)^{\frac{\rho^2-\rho}{2}}.$$

This proves the theorem. □

Note that this success probability is 0 for $\rho = 0$ and is monotonically increasing. As a function of ρ it presents itself as an S-shaped curve with a steep slope at approximately $\rho = 2^{\frac{n}{2}}$ (see Figure 3(a) for an example). Thus, for the authentication to be successful with an overwhelming probability, the size ρ of $\mathcal{P}$'s

[2] To simplify the discussion, we sample from the whole set $\{0,1\}^n$ instead of $\{0,1\}^n \setminus \{0^n\}$. This does not affect the overall analysis, since the value 0^n occurs with a negligible probability.

database $\mathcal{D}$ should be chosen right after this steep slope, ensuring a probability close to 1. To give the reader an idea on how the database size ρ behaves in practice, we state that sizes of about $\rho \approx 2^{17}$ or $\rho \approx 2^{25}$ are realistic in most real-world applications. Details on this and other numerical examples can be found in Section 5.[3]

3.3 Authenticated Key Exchange

A minor modification of our authentication protocol yields an authenticated key exchange between the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ (here, "authenticated" refers to the verifier $\mathcal{V}$ only, since the authentication in our protocol is unilateral). More precisely we will achieve that, if the authentication of $\mathcal{P}$ is successful, both $\mathcal{V}$ and $\mathcal{P}$ compute the same shared secret key K. If the authentication fails, $\mathcal{P}$ computes a random key, while $\mathcal{V}$ computes the "correct" key K. These two keys will be the same with a probability that is negligible in n.

Next, we describe the modification of our protocol: Let $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^{2n}$ be a (publicly known) cryptographic hash function. Now, the only modifications we make to our authentication protocol are (see key computation in square brackets in Fig. 2):

1. After the prover $\mathcal{P}$ created the two PUF-challenges C_1 and C_2 together with the corresponding helper data h_1 and h_2, respectively, he computes the *key* $K = \mathcal{H}(R(C_1, h_1) \| R(C_2, h_2))$.
2. After the verifier $\mathcal{V}$ checked the authenticity of $\mathcal{P}$ and computed the message M, he computes the *key* $K = \mathcal{H}(R(C_1, h_1) \| R(C_2, h_2))$.

It can be seen immediately (when $\mathcal{H}$ is again modelled as a random oracle) that if the authentication of $\mathcal{P}$ fails, $\mathcal{P}$ will compute a key K that is uniformly distributed in $\{0,1\}^{2n}$. Therefore, the probability that $\mathcal{P}$'s key and $\mathcal{V}$'s key coincide is 2^{-2n} which is negligible in n. Otherwise, both parties have exchanged a secret key.

4 Security Model and Analysis

The security model for our authentication protocol considers a *passive* adversary $\mathcal{A}$ only.[4] This means that the adversary $\mathcal{A}$ is only able to passively listen on the communication channel, and does neither have access to the underlying PUF,

[3] We stress that the generation of a database of size 2^{25} in the enrolment phase is *not* impractical. The reason for this is that the enrolment phase is not carried out with the actual resource-constrained verifier but in a trusted environment. In particular, this means that the database is generated before the Controlled-PUF is engineered into the verifier-device.

[4] Our authentication protocol does not rely on a confidential communication channel. All messages are being sent in the clear. It is easy to see that, when considering an *active* adversary $\mathcal{A}$ that can for instance manipulate these messages, the authentication will fail with overwhelming probability.

nor can do any invasive analysis on the used components. More precisely, $\mathcal{A}$ is allowed to see a bounded number of protocol transcripts (a transcript is a copy of all messages sent by the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ after a complete run of the authentication protocol), and then tries to break the protocol. Here, breaking the protocol means that $\mathcal{A}$ can successfully authenticate herself to the verifier $\mathcal{V}$. We briefly recall that a successful authentication amounts to finding two PUF-challenges C_1, C_2 with helper data h_1, h_2 such that for a given $\Delta \xleftarrow{U} \{0,1\}^n$,[5] the corresponding responses (after applying the hash function H and the reconstruction phase of the FE to the PUF's outputs) satisfy that $R(C_1, h_1) \oplus R(C_2, h_2) = \Delta$. Formally, the security of our protocol is modelled as follows:

Definition 1. *Let κ denote the (bit-) entropy of the output of the reconstruction phase of the FE in our authentication protocol. Then, our authentication protocol is called* (t, κ, ε)-secure (against passive adversaries)*, if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ who gets to see t* transcripts $\tau_i = (\Delta_i, (C_i, C_i'), (h_i, h_i'))$*, where $\Delta_i = R(C_i, h_i) \oplus R(C_i', h_i')$, for $i = 1, \ldots, t$, successfully authenticates herself with probability at most ε, i.e.,*

$$\Pr\left[\mathcal{A}(\tau_1, \ldots, \tau_t) = ((C, C'), (h, h')) \mid \Delta = R(C, h) \oplus R(C', h')\right] \leq \varepsilon,$$

where the probability is taken over the random coin tosses of $\mathcal{A}$ and random $\Delta \xleftarrow{U} \{0,1\}^n$. We denote this success probability of $\mathcal{A}$ by $\mathsf{Succ}_{\mathcal{A},n,\kappa}(t)$.

This section deals with the question of how many protocol transcripts τ the adversary $\mathcal{A}$ has to see *at least* in order to successfully authenticate herself with some prescribed probability p. In other words, we will derive a formula that computes the success probability $\mathsf{Succ}^{\mathrm{wc}}_{\mathcal{A},n,\kappa}(t)$ of a *worst-case* adversary $\mathcal{A}$ that gets to see t transcripts. Before we do so, we need to clarify what the worst-case scenario is. To this end, we first show that since an adversary $\mathcal{A}$ never sees neither the PUF-responses nor the actual outputs of the complete construction (i.e., after applying the hash function and the FE to the PUF's outputs), the helper data h that is included in each transcript τ is completely independent of the PUF-challenges C and hence is of *no* use to $\mathcal{A}$.

On the Inutility of Helper Data. We assume that the underlying PUF produces at least 2^κ many different responses. The only relation of the helper data to the PUF-challenges is the value Δ and the PUF-responses (which the adversary $\mathcal{A}$ never sees): By construction, we have that $R(C, h) = H(\gamma)$ and $R(C', h') = H(\gamma')$ (with $H(\gamma) \oplus H(\gamma') = \Delta$) where γ, γ' are outputs of the reconstruction phase of the FE each having κ bits of entropy. Since we assume that the adversary $\mathcal{A}$ does not know the behaviour of the used PUF, she does not have any information about the PUF-responses R and R' of C and C', respectively. But for each helper data h, there are at least 2^κ different PUF-responses R that,

[5] We include the zero element 0^n as possible Δ-value, since it occurs with negligible probability but simplifies the discussion in this section.

together with the helper data h, will lead to a valid γ in the reconstruction phase of the FE. Then in turn, checking which γ is the correct one, the adversary $\mathcal{A}$ first needs to compute $H(\gamma)$ (and analogously $H(\gamma')$) and then check whether $H(\gamma) \oplus H(\gamma') = \Delta$. Since the hash function H is modelled as a random oracle, the best $\mathcal{A}$ can do is to fix the first hash value $H(\gamma)$ and then try all 2^κ many γ' (brute-force) or to guess this value, which is successful with probability 2^κ. Obviously, we can do the same discussion with randomly chosen helper data, which shows that the helper data is indistinguishable (in the parameter κ) from random to $\mathcal{A}$.

The Worst-Case Scenario. After seeing t transcripts, the adversary $\mathcal{A}$ has a database of t (not necessarily different) tuples of the form $(\Delta, (C, C'), (h, h'))$ such that $R(C, h) \oplus R(C', h') = \Delta$. We emphasize that $\mathcal{A}$ does not know the actual values $R(C, h)$. Now, the previous discussion allows us to forget about the helper data part in $\mathcal{A}$'s database, as it does not give the adversary any additional information (from now on, we will write $R(C)$ instead of $R(C, h)$). Then in turn, we can think of $\mathcal{A}$'s database as being a list of $2t$ PUF-challenges $C_1, \ldots, C_{2t}$ where $\mathcal{A}$ knows for at least t pairs the value $R(C_i) \oplus R(C_j) = \Delta_{i,j}$.

We consider the following example and assume that one of the PUF-challenges is always fixed, say to C_1. Then, after seeing t transcripts, the adversary $\mathcal{A}$ gets the following system of t equations:

$$R(C_1) \oplus R(C_j) = \Delta_{1,j} \text{ for all } j = 2, \ldots, t+1.$$

Adding any two of these yields a new equation of the form $R(C_i) \oplus R(C_j) = \Delta_{i,j}$ for $2 \leq i < j \leq t+1$. This means that the adversary can construct up to $\binom{t}{2} - t$ additional Δ-values that she has not seen before in any of the transcripts. Note that this is all an adversary can do, since the challenges and the values Δ are chosen uniformly at random and the PUF is unpredictable. Moreover, if one of these Δ's is challenged in an authentication, the adversary can check whether she can construct it from her known PUF-challenges in her database. We therefore call such Δ-values $\mathcal{A}$-*checkable*.

With this example in mind, we see that the *worst-case scenario* (which is the best case for the adversary) occurs, when there are exactly $\binom{t}{2}$ $\mathcal{A}$-checkable Δ-values. On the other hand, there are only 2^n different Δ-values in total, so if $\binom{t}{2} = \frac{t^2-t}{2} = 2^n$, all Δ-values are $\mathcal{A}$-checkable and the adversary can successfully authenticate with probability 1. This equation, however, is satisfied if and only if t is a positive root of the degree 2 polynomial $X^2 - X - 2^{n+1}$, which in turn is satisfied if and only if $t = \frac{1}{2} + \frac{1}{2}\sqrt{1 + 2^{n+3}}$ by using the "quadratic formula". This means that once the adversary $\mathcal{A}$ has seen more than $t = \left\lfloor \frac{1}{2} + \frac{1}{2}\sqrt{1 + 2^{n+3}} \right\rfloor$ transcripts, she can successfully authenticate herself with probability 1 in the worst-case scenario.

Security Analysis. Having clarified what the worst-case scenario is, considering a passive adversary $\mathcal{A}$, we can finally prove the main theorem of this section:

Theorem 2. *Our authentication protocol is* $(t, \kappa, \mathsf{Succ}^{\mathrm{wc}}_{\mathcal{A},n,\kappa}(t))$*-secure, where*

$$\mathsf{Succ}^{\mathrm{wc}}_{\mathcal{A},n,\kappa}(t) = \begin{cases} 1 & , \text{ if } t > \left\lfloor \frac{1}{2} + \frac{1}{2}\sqrt{1+2^{n+3}} \right\rfloor \\ \frac{(2^\kappa-1)t^2-(2^\kappa-1)t+2^{n+1}}{2^{n+\kappa+1}} & , \text{ else} \end{cases}$$

is the probability of a worst-case adversary $\mathcal{A}$ *successesfully authenticating herself after having seen* t *transcripts, and* κ *is the (bit-) entropy of the FE's output.*

Proof. Let $\mathcal{B}$ be an arbitrary PPT adversary on our authentication protocol. Since $\mathcal{A}$ is a worst-case adversary, we have that $\mathsf{Succ}_{\mathcal{B},n,\kappa}(t) \leq \mathsf{Succ}^{\mathrm{wc}}_{\mathcal{A},n,\kappa}(t)$. So by Definition 1, it suffices to compute $\mathsf{Succ}^{\mathrm{wc}}_{\mathcal{A},n,\kappa}(t)$. Right above the Theorem, we have already shown that $\mathsf{Succ}^{\mathrm{wc}}_{\mathcal{A},n,\kappa}(t) = 1$, if $t > \left\lfloor \frac{1}{2} + \frac{1}{2}\sqrt{1+2^{n+3}} \right\rfloor$ by using the "quadratic formula" to find a positive root of $X^2 - X - 2^{n+1}$.

On the other hand, if $t \leq \left\lfloor \frac{1}{2} + \frac{1}{2}\sqrt{1+2^{n+3}} \right\rfloor$, i.e., $\binom{t}{2} \leq 2^n$, we know that there are precisely $\binom{t}{2}$ $\mathcal{A}$-checkable Δ-values, by definition of the worst-case scenario. So for a given random challenge $\Delta \xleftarrow{U} \{0,1\}^n$ (when $\mathcal{A}$ is trying to authenticate herself), the probability that we hit on one of these $\mathcal{A}$-checkable Δ-values is $\frac{\binom{t}{2}}{2^n} = \frac{t^2-t}{2^{n+1}}$, i.e.,

$$\Pr_{\Delta \xleftarrow{U} \{0,1\}^n}[\Delta \text{ is } \mathcal{A}\text{-checkable}] = \frac{t^2-t}{2^{n+1}}.$$

Then again, if we hit on a Δ that is not $\mathcal{A}$-checkable, we know by definition of the worst case that it cannot be the XOR of two responses to values in $\mathcal{A}$'s database at all. This is because if there are precisely $\binom{t}{2}$ many $\mathcal{A}$-checkable Δ-values, the adversary $\mathcal{A}$ can only construct precisely t linearly dependent equations from the t transcripts she has seen. However, this means that there are $\binom{t}{2}$ many Δ-values that can be constructed as the XOR of two responses to values in $\mathcal{A}$'s database. But since there are precisely $\binom{t}{2}$ many $\mathcal{A}$-checkable Δ-values, these must have been all such values.

Now that we know the probability of hitting on an $\mathcal{A}$-checkable Δ-value, we also know the probability of not hitting on one, namely:

$$\Pr_{\Delta \xleftarrow{U} \{0,1\}^n}[\Delta \text{ is not } \mathcal{A}\text{-checkable}] = 1 - \frac{t^2-t}{2^{n+1}} = \frac{2^{n+1}-t^2+t}{2^{n+1}}.$$

In such a case though, the adversary $\mathcal{A}$ cannot do better than guessing two PUF-challenges C_1, C_2 (and actually some random helper data that we neglect here, although it would actually reduce the success probability of $\mathcal{A}$ even more). But the probability of guessing correctly (meaning that $R(C_1) \oplus R(C_2) = \Delta$) is upper bounded by the probability of guessing two outputs γ_1, γ_2 of the FE such that $H(\gamma_1) \oplus H(\gamma_2) = \Delta$, which is $\frac{1}{2^\kappa}$ where κ is the (bit-) entropy of the outputs of the FE. So if Δ is not $\mathcal{A}$-checkable, the success probability of $\mathcal{A}$ is less or equal to $\frac{2^{n+1}-t^2+t}{2^{n+1}} \cdot \frac{1}{2^\kappa}$.

In total, this shows that if $t \leq \left\lfloor \frac{1}{2} + \frac{1}{2}\sqrt{1+2^{n+3}} \right\rfloor$, $\mathcal{A}$'s probability of successfully authenticating herself is upper bounded by

$$\frac{(2^\kappa - 1)t^2 - (2^\kappa - 1)t + 2^{n+1}}{2^{n+\kappa+1}}.$$

This completes the proof. □

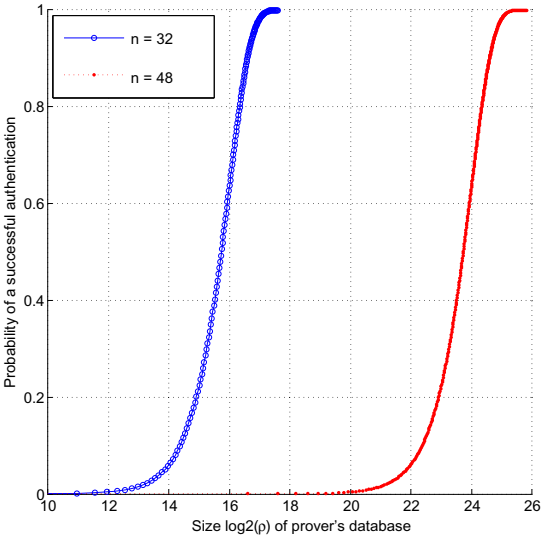
We stress that by considering a worst-case adversary, the probability in Theorem 2 is overly pessimistic since the described worst-case scenario does happen with a very small probability only. Furthermore, we want to mention that in many existing authentication schemes, a passive adversary can perform model-building attacks on the used PUF [25]. This is done by collecting a subset of all CRPs, and then trying to create a mathematical model that allows emulating the PUF in software. However, for this attack to work, the adversary needs to have access to the PUF's responses. We counter this problem in our protocol by using a controlled PUF which hides the actual PUF responses from the adversary. This way of protecting protocols against model-building attacks is well-known and also mentioned in [25].

Replay Attacks. We stress that our above worst-case analysis captures replay attacks as well. In fact, by the birthday paradox, the probability of a successful replay attack (after having seen t transcripts) equals $1 - e^{-\frac{t^2}{2^{n+1}}}$. But this term grows more slowly than $\mathsf{Succ}^{\mathrm{wc}}_{\mathcal{A},n,\kappa}(t)$ and is always smaller than this for relevant sizes of t. For realistic values, such as $n = 32$ and $\kappa = 48$ (cf. Section 5), the probability of a successful replay attack is *always* smaller than $\mathsf{Succ}^{\mathrm{wc}}_{\mathcal{A},n,\kappa}(t)$ when the adversary has seen more than $t = 2581$ transcripts. But even if the adversary sees $t = 9268$ transcripts, this probability is still to small to raise any realistic security concerns.

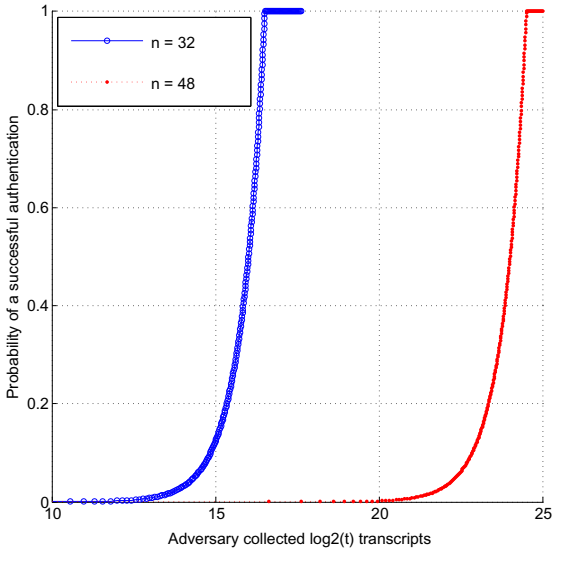
5 Instantiation of the Protocol

In this section, we give a concrete instantiation of our authentication protocol which involves choosing appropriate PUFs, Fuzzy Extractors, Random Number Generators, and hash functions. Starting with the first of these, we note that we will use Arbiter PUFs which, according to [17], have a bit error rate ℓ of 3%. We stress again that our authentication protocol hides the PUF-responses, so the existing model-building attacks [25] do not work. Based on the PUF's error rate, we choose a binary linear $[\mu, k, d]$-code that can correct at least the errors that the PUF produces, for the Fuzzy Extractor. In practice, we use a certain Golay code from [3] for this implementation. An example step-by-step implementation is as follows:

1. Fix a desired output length n of the controlled PUF and the desired entropy κ we want the FE to have – these lengths basically are the security parameters as they determines the amount of protocol transcripts a worst-case adversary is allowed to see before she can break the protocol with a prescribed success



(a)



(b)

Fig. 3. (a) Probability $\mathsf{Succ}^{\mathrm{Auth}}_{\mathcal{P},n}(\rho)$ of a successful authentication for varying sizes ρ of $\mathcal{P}$'s database $\mathcal{D}$ and fixed values $n = 32$ and $n = 48$. (b) Success probability $\mathsf{Succ}^{\mathrm{wc}}_{\mathcal{A},n,\kappa}(t)$ of a worst-case adversary $\mathcal{A}$ for a growing number of protocol transcripts t she has seen and fixed values $n = 32$ and $n = 48$, while $\kappa = 48$. Note the logarithmic x-axis.

probability (cf. Theorem 2). Here, we fix $n = 32$, $\kappa = 48$ and want to bound the success probability by 0.01. As an alternative, we also provide the case where $n = 48$ for the same $\kappa = 48$.
2. Choose a cryptographic hash function $H : \{0,1\}^* \longrightarrow \{0,1\}^n$. Here, we use an LFSR-based Toeplitz Hash function of output length 32 (cf. [9]). In our alternative parameter setting, we need an output length of 48 bits.
3. Choose κ Arbiter PUFs – this ensures precisely 2^κ many PUF-responses.
4. Choose a binary linear $[\mu, k, d]$-code $\mathcal{C}$ which can correct at least $\frac{\ell \cdot \kappa}{100}$ errors. Here, we choose a $[23, 12, 7]$-Golay code (from [3]) which can correct up to $3 \geq \frac{3 \cdot 48}{100} \approx 2$ errors. In order to get an entropy of $\kappa = 48$ bits in the FE's output, we divide an output of the PUF into 4 parts containing 12 bits each. Then, we append 11 zeros to each part to ensure a length of 23. After this we continue with the usual protocol, which means that we have to use the reconstruction phase of the FE 4 times and create 4 helper data. In each authentication round, the prover then needs to send 4 helper data instead of just 1. As we have shown in Section 4, this does not affect the security of our scheme. The reconstruction phase of the FE also needs to run 4 times which creates 4 code words $\gamma_1, \ldots, \gamma_4$ of length 23 containing 12 bits of entropy each. The final evaluation of the hash function H will then be on the concatenation of these 4 code words, i.e., $H(\gamma_1 \| \ldots \| \gamma_4)$. We notice that the input $\gamma_1 \| \ldots \| \gamma_4$ to H has 48 bits of entropy which means that in Theorem 2, we can use the paramater $\kappa = 48$ as we desired.

According to Theorem 1, when our protocol is instatiated with these parameters where $n = 32, \kappa = 48$ (or $n = 48$), the prover $\mathcal{P}$'s database $\mathcal{D}$ can be constructed to have size $\rho = 140639$ (or $\rho = 36003337$) which ensures a successful authentication with probability $\mathsf{Succ}^{\mathrm{Auth}}_{\mathcal{P},n}(\rho) \geq 0.99$ (or $\mathsf{Succ}^{\mathrm{Auth}}_{\mathcal{P},n}(\rho) \geq 0.99$), cf. Fig. 3(a). Concerning the security of our protocol in this instantiation, Fig. 3(b) tells us that a worst-case adversary is allowed to see at most $t = 9268$ (or $t = 2372657$) protocol transcripts to ensure a success probability $\mathsf{Succ}^{\mathrm{wc}}_{\mathcal{A},n,\kappa}(t) < 0.01$ (or $\mathsf{Succ}^{\mathrm{wc}}_{\mathcal{A},n,\kappa}(t) < 0.01$), cf. Theorem 2.

Depending on the application scenario, we can arbitrarily vary the above parameters in order to get a higher level of security but on the cost of efficiency.

6 Conclusion

Motivated by the fact that previous PUF-based authentication protocols fail to work in some application scenarios, we introduced the new notion of *converse* PUF-based authentication: opposed to previous solutions, in our approach the verifier holds a PUF while the prover does not. We presented the first such protocol, gave an extensive security analysis, and showed that it can also be used for authenticated key exchange. Besides the mentioned application examples in the present paper, future work includes the employment of our new protocol to other applications. Additionally, we consider an actual implementation on resource-constraint devices (such as sensor nodes) as an interesting work to pursue.

Acknowledgement. This work has been supported in part by the European Commission through the FP7 programme under contract 238811 UNIQUE.

References

1. Armknecht, F., Maes, R., Sadeghi, A.-R., Sunar, B., Tuyls, P.: Memory Leakage-Resilient Encryption Based on Physically Unclonable Functions. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 685–702. Springer, Heidelberg (2009)
2. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 1993, pp. 62–73. ACM (1993)
3. Bösch, C., Guajardo, J., Sadeghi, A.-R., Shokrollahi, J., Tuyls, P.: Efficient Helper Data Key Extractor on FPGAs. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 181–197. Springer, Heidelberg (2008)
4. Bringer, J., Chabanne, H., Icart, T.: On Physical Obfuscation of Cryptographic Algorithms. In: Roy, B., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 88–103. Springer, Heidelberg (2009)
5. Das, A., Kocabaş, Ü., Sadeghi, A.-R., Verbauwhede, I.: PUF-based Secure Test Wrapper Design for Cryptographic SoC Testing. In: Design, Automation and Test in Europe (DATE). IEEE (2012)
6. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 523–540. Springer, Heidelberg (2004)
7. Eichhorn, I., Koeberl, P., van der Leest, V.: Logically reconfigurable PUFs: Memory-based secure key storage. In: ACM Workshop on Scalable Trusted Computing (ACM STC), pp. 59–64. ACM, New York (2011)
8. Gassend, B.: Physical Random Functions. Master's thesis, MIT, MA, USA (January 2003)
9. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Controlled physical random functions. In: Computer Security Applications Conference (ACSAC), pp. 149–160. IEEE (2002)
10. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Controlled physical random functions. In: Computer Security Applications Conference (ACSAC), pp. 149–160. IEEE (2002)
11. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002), pp. 148–160. ACM (2002)
12. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: ACM Conference on Computer and Communications Security (ACM CCS), pp. 148–160. ACM, New York (2002)
13. Guajardo, J., Kumar, S.S., Schrijen, G.-J., Tuyls, P.: FPGA Intrinsic PUFs and Their Use for IP Protection. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 63–80. Springer, Heidelberg (2007)
14. Guajardo, J., Kumar, S.S., Schrijen, G.-J., Tuyls, P.: Brand and IP protection with physical unclonable functions. In: IEEE International Symposium on Circuits and Systems (ISCAS) 2008, pp. 3186–3189. IEEE (May 2008)
15. Holcomb, D.E., Burleson, W.P., Fu, K.: Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In: Conference on RFID Security 2007, Malaga, Spain, July 11-13 (2007)
16. Kumar, S.S., Guajardo, J., Maes, R., Schrijen, G.-J., Tuyls, P.: Extended abstract: The butterfly PUF protecting IP on every FPGA. In: Workshop on Hardware-Oriented Security (HOST), pp. 67–70. IEEE (June 2008)

17. Lee, J.W., Lim, D., Gassend, B., Suh, E.G., van Dijk, M., Devadas, S.: A technique to build a secret key in integrated circuits for identification and authentication applications. In: Symposium on VLSI Circuits, pp. 176–179. IEEE (June 2004)
18. Lim, D., Lee, J.W., Gassend, B., Suh, E.G., van Dijk, M., Devadas, S.: Extracting secret keys from integrated circuits. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 13(10), 1200–1205 (2005)
19. Lin, L., Holcomb, D., Krishnappa, D.K., Shabadi, P., Burleson, W.: Low-power sub-threshold design of secure physical unclonable functions. In: International Symposium on Low-Power Electronics and Design (ISLPED), pp. 43–48. IEEE (August 2010)
20. Maes, R., Tuyls, P., Verbauwhede, I.: Intrinsic PUFs from flip-flops on reconfigurable devices (November 2008)
21. Maes, R., Verbauwhede, I.: Physically unclonable functions: A study on the state of the art and future research directions. In: Towards Hardware-Intrinsic Security (2010)
22. Maiti, A., Casarona, J., McHale, L., Schaumont, P.: A large scale characterization of RO-PUF. In: International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 94–99. IEEE (June 2010)
23. Öztürk, E., Hammouri, G., Sunar, B.: Towards robust low cost authentication for pervasive devices. In: International Conference on Pervasive Computing and Communications (PerCom), pp. 170–178. IEEE, Washington, DC (2008)
24. Ranasinghe, D.C., Engels, D.W., Cole, P.H.: Security and privacy: Modest proposals for Low-Cost RFID systems. In: Auto-ID Labs Research Workshop (September 2004)
25. Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., Schmidhuber, J.: Modeling attacks on physical unclonable functions. In: ACM Conference on Computer and Communications Security (ACM CCS), pp. 237–249. ACM, New York (2010)
26. Sadeghi, A.-R., Visconti, I., Wachsmann, C.: Enhancing RFID security and privacy by physically unclonable functions. In: Towards Hardware-Intrinsic Security. Information Security and Cryptography, pp. 281–305. Springer, Heidelberg (2010)
27. Schulz, S., Sadeghi, A.-R., Wachsmann, C.: Short paper: Lightweight remote attestation using physical functions. In: Proceedings of the Fourth ACM Conference on Wireless Network Security (ACM WiSec), pp. 109–114. ACM, New York (2011)
28. Su, Y., Holleman, J., Otis, B.P.: A digital 1.6 pJ/bit chip identification circuit using process variations. IEEE Journal of Solid-State Circuits 43(1), 69–77 (2008)
29. Suh, E.G., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: ACM/IEEE Design Automation Conference (DAC), pp. 9–14. IEEE (June 2007)
30. Tuyls, P., Batina, L.: RFID-Tags for Anti-counterfeiting. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 115–131. Springer, Heidelberg (2006)
31. Tuyls, P., Schrijen, G.-J., Škorić, B., van Geloven, J., Verhaegh, N., Wolters, R.: Read-Proof Hardware from Protective Coatings. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 369–383. Springer, Heidelberg (2006)
32. van der Leest, V., Schrijen, G.-J., Handschuh, H., Tuyls, P.: Hardware intrinsic security from D flip-flops. In: ACM Workshop on Scalable Trusted Computing (ACM STC), pp. 53–62. ACM, New York (2010)
33. Škorić, B., Tuyls, P., Ophey, W.: Robust Key Extraction from Physical Uncloneable Functions. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 407–422. Springer, Heidelberg (2005)