Secure Lightweight Entity Authentication with Strong PUFs: Mission Impossible II

Jeroen Delvaux[1,2], Dawu Gu[2], Roel Peeters[1] and Ingrid Verbauwhede[1]

[1] ESAT/COSIC and iMinds, KU Leuven,
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
{*jeroen.delvaux, roel.peeters, ingrid.verbauwhede*}*@esat.kuleuven.be*
[2] CSE/LoCCS, Shanghai Jiao Tong University,
800 Dongchuan Road, Shanghai 200240, China
dwgu@sjtu.edu.cn

Abstract. Physically unclonable functions (PUFs) exploit the unavoidable manufacturing variations of an integrated circuit (IC). Their input-output behavior serves as a unique IC 'fingerprint'. Therefore, they have been envisioned as an IC authentication mechanism, in particular the subclass of so-called strong PUFs. The protocol proposals are typically accompanied with two PUF promises: lightweight and an increased resistance against physical attacks. In our prior CHES 2014 manuscript, we reviewed eight proposals in chronological order. This work comprehends a sequel. Most notably, five additional strong PUF protocols are included in our large-scale overview-analysis. Again, numerous security and practicality issues are revealed. Furthermore, we improve the transparency of our analysis by explicitly listing protocol requirements. These can also be used as a guideline for future protocol design. Finally, token privacy has been included in the analysis.

Keywords: physically unclonable function, entity authentication, lightweight

1 Introduction

In this work, we consider a common authentication scenario between two parties: a **low-cost resource-constrained token** and a **resource-rich server**. Practical instantiations of a token include the following: RFID tags, smart cards and nodes of a wireless sensor network. One-way or possibly mutual entity authentication is the main objective, occasionally extended with token privacy. Although the server has secure computing and storage at its disposal, providing security is a major challenge given the requirements at the token side. Tokens typically store a secret key in non-volatile memory (NVM), using a mature technology such as EEPROM and its successor Flash, battery-backed SRAM or fuses. Cryptographic primitives import the key and perform an authentication protocol.

Today's issues are as follows. First, implementing cryptographic primitives in a resource-constrained manner is rather challenging. A lightweight hash function (Spongent, Quark, Photon, etc.) still requires 1500 GE, typically. Second, an attacker can gain physical access to the integrated circuit (IC) of a token. NVM has proven to be vulnerable to physical attacks [28] as the secret is stored permanently in a robust electrical manner. Third, most NVM technologies oppose the low-cost objective. EEPROM/Flash requires floating gate transistors, resulting in additional manufacturing steps with respect to a regular CMOS design. Battery-backed SRAM requires a battery. Circuitry to protect the NVM contents, e.g. a mesh of sensing wires, tends to be expensive.

Physically unclonable functions (PUFs) offer a promising alternative. Essentially, they are binary functions, with their input-output behavior determined by IC manufacturing variations. Therefore, they can be understood as a unique IC 'fingerprint', analogous to human biometrics. They might alleviate the aforementioned issues. Many PUFs allow for an implementation which is both resource-constrained and CMOS compatible. Furthermore, the secret is hidden in the physical structure of an IC, which is a much less readable format. Invasive attacks might easily destroy this secret, as an additional advantage.

Several PUF-based **authentication protocols** have been proposed, in particular for the subclass of so-called strong PUFs. In our prior CHES 2014 manuscript [1], we reviewed the following eight proposals: basic strong PUF authentication [23], controlled PUFs [4], Öztürk et al. [22], Hammouri et al. [6], logically

reconfigurable PUFs [12], reverse fuzzy extractors [31], the converse protocol [13] end slender PUFs [20]. In this work, we extend our analysis with five additional protocols: Kulseng et al. [15] Jin et al. [10], Xu et al. [32], noise bifurcation [33] and system of PUFs [14].

Similar to key-storage in NVM, all protocols in this work employ **two phases**. The first phase is a one-time enrollment in a secure environment, following IC manufacturing. The server then obtains some information about the PUF to establish a shared secret. The destruction of a one-time interface might permanently restrict direct PUF access afterwards. The second phase is in-the-field deployment, where tokens are vulnerable to physical attacks. Token and server then authenticate over an insecure communication channel.

The remainder of this paper is organized as follows. Section 2 introduces notation and preliminaries. Section 3 lists the requirements of a PUF-based authentication protocol. Section 4 describes and analyzes all strong PUF protocols. Section 5 provides an overview of the protocol issues. Section 6 concludes the work. Our analysis is performed at protocol level, considering PUFs as a black box. The occasional need for PUF internals is provided by Appendix A in [1].

2 Preliminaries

2.1 Notation

Binary vectors are denoted with a bold lowercase character, e.g. $\mathbf{c} \in \{0,1\}^{1\times m}$. All vectors are row vectors. Their elements are selected with an index $i \geq 1$ between round brackets, e.g. $\mathbf{c}(1)$, $\mathbf{c}(2)$, etc. The null vector is denoted as $\mathbf{0}$. Binary matrices are denoted with a bold uppercase character, e.g. $\mathbf{H}$. Operations are the following: addition modulo 2 (XOR), e.g. $\mathbf{x} \oplus \mathbf{c}$, multiplication modulo 2, e.g. $\mathbf{e} \cdot \mathbf{H}^T$, concatenation, e.g. $\mathbf{x}||\mathbf{c}$, and bit inversion, e.g. $\bar{\mathbf{r}}$. Variable assignment is denoted with an arrow, e.g. $d \leftarrow d-1$. Variables may occasionally be denoted with an additional protocol run counter, e.g. $\mathbf{c}^{(1)}$, $\mathbf{c}^{(2)}$, etc. Functions are printed in *italic*, with their input arguments between round brackets, e.g. Hamming weight $HW(\mathbf{r})$ and Hamming distance $HD(\mathbf{r}_1, \mathbf{r}_2)$.

2.2 Physically Unclonable Functions: Black Box Description

The m-bit input and n-bit output of a PUF are referred to as challenge $\mathbf{c}$ and response $\mathbf{r}$ respectively. Unfortunately for cryptographic purposes, the behavior of challenge-response pairs (CRPs) does not correspond with a random oracle. First, the **response bits are not perfectly reproducible**. Noise in transistors and wires as well as various environmental perturbations (supply voltage, temperature, etc.) result in non-determinism. Second, the response bits are **non-uniformly distributed**: bias and correlations are present. Without proper compensation, this might enable a variety of attacks.

PUFs are often subdivided in two classes, according to their number of CRPs. **Weak PUFs** offer few CRPs, scaling roughly linear with the required IC area. Architectures typically consist of an array of identically laid-out cells, each producing one or more response bits. E.g. the SRAM PUF [9] and the ring oscillator PUF[1] [29] are both very popular. Addressing the array provides a challenge-response mechanism. However, the total array size is of primary importance: mostly, all 2^{m+n} bits are used collectively to generate a secret key. This key is stored in volatile memory whenever cryptographic operations have to be performed, opposing the permanent nature of NVM. Post-processing logic, typically a fuzzy extractor [3], is required to ensure a reproducible and uniformly distributed key.

Strong PUFs offer an enormous number of CRPs, often scaling exponentially with the required IC area. Despite their small response space, mostly $n = 1$, architectures are typically able to provide a large challenge space, e.g. $m = 128$. Therefore, they might greatly exceed the need for secret key generation and have been promoted primarily as lightweight authentication primitives. The most famous example is the arbiter PUF [16]. However, due to inherent correlations, unprotected exposure to the PUF might enable so-called modeling attacks. One tries to construct a predictive model of the PUF, given a limited set of training CRPs. Machine learning algorithms have proven to be successful [26]. For completeness, we note that the definition of strong PUFs has shifted over the years. The original more specific notion in [5] assumes a large response space in addition to strong cryptographic properties: resistance against modeling and tamper evidence. Although highly relevant as an ideal case specification, we stick to the more recent practical notion.

[1] We consider the most usable read-out modes which aim to avoid correlations, e.g. pairing neighboring oscillators.

2.3 Secure Sketches and Fuzzy Extractors

The noisiness of a PUF causes the regenerated instance of a response $\mathbf{r}$ to be slightly different: $\widetilde{\mathbf{r}} = \mathbf{r} \oplus \mathbf{e}$, with $HW(\mathbf{e})$ relatively small. Secure sketches [3] are a useful reconstruction tool, as defined by a two-step procedure. First, public helper data is generated: $\mathbf{p} = Gen(\mathbf{r})$. Second, reproduction is performed: $\mathbf{r} = Rep(\widetilde{\mathbf{r}}, \mathbf{p})$. Helper data $\mathbf{p}$ unavoidably leaks some information about $\mathbf{r}$, although this entropy loss is supposed to be limited. Despite the rather generic definition, two constructions dominate the implementation landscape, as specified below. Both the code-offset and syndrome construction employ a binary $[n, k, t]$ block code $\mathcal{C}$, with t the error-correcting capability. The latter construction requires a linear block code, as it employs the parity check matrix $\mathbf{H} \in \{0,1\}^{(n-k)\times n}$. Successful reconstruction is guaranteed for both constructions, given $HW(\mathbf{e}) \leq t$. Information leakage is limited to $n-k$ bits. The hardware footprint is asymmetric: *Gen* is better suited for resource-constrained devices than *Rep* [31]. A fuzzy extractor (FE) can be constructed out of a secure sketch. An additional hash function then produces a nearly uniform output: $\mathbf{k} \leftarrow Hash(\mathbf{r})$. By having more input than output bits, it compensates for the non-uniformity of $\mathbf{r}$ as well as the helper data leakage.

code-offset	*Gen*	*Rep*
	Random $\mathbf{w} \in \mathcal{C}$	$\widetilde{\mathbf{w}} \leftarrow \widetilde{\mathbf{r}} \oplus \mathbf{p} = \mathbf{w} \oplus \mathbf{e}$
	$\mathbf{p} \leftarrow \mathbf{r} \oplus \mathbf{w}$	Error-correct $\widetilde{\mathbf{w}}$ to $\mathbf{w}$
		$\mathbf{r} \leftarrow \mathbf{p} \oplus \mathbf{w}$

syndrome	*Gen*	*Rep*
	$\mathbf{p} \leftarrow \mathbf{r} \cdot \mathbf{H}^T$	$\mathbf{s} \leftarrow \widetilde{\mathbf{r}} \cdot \mathbf{H}^T \oplus \mathbf{p} = \mathbf{e} \cdot \mathbf{H}^T$
		Determine $\mathbf{e}$
		$\mathbf{r} \leftarrow \widetilde{\mathbf{r}} \oplus \mathbf{e}$

3 Protocol Requirements

PUFs require a special flavor of protocol design. We are the first to explicitly list an extensive set of protocol requirements. Sections 3.1 to 3.5 describe the PUF-induced requirements. Sections 3.7 to 3.9 cover more conventional concerns, which of course still apply.

3.1 Benefit with Respect to NVM (#1)

Advantages of PUF technology with respect to NVM should be preserved. This is primarily low-cost manufacturing and improved physical security. Otherwise, one could equally well opt for traditional key-storage. To illustrate further, PUF behavior could in principle be mimicked with the latter. Secure NVM allows to instantiate an IC-specific secret function, with the lack of noise as additional advantage. We define an NVM-based authentication protocol to highlight this requirement, further denoted as Reference I. Each token stores a unique secret key $\mathbf{k}$, as represented by Figures 1(a) and 1(b). Additional cryptographic logic performs the authentication. A block cipher would be an excellent choice. However, for ease of comparison, we prefer a hash function. Reference I-A provides token authenticity only. The server checks whether a token can compute $\mathbf{a} \leftarrow Hash(\mathbf{k}, \mathbf{n})$, with $\mathbf{n}$ a random nonce. Reference I-B provides server authenticity only. A token checks whether the server can compute $\mathbf{b} \leftarrow Hash(\mathbf{k}, \mathbf{n})$, with $\mathbf{n}$ a random nonce. The latter requires an on-chip true random number generator (TRNG). A mixture of both mechanisms could provide mutual authentication.

3.2 Able to Handle Noisiness (#2)

Noisiness of the PUF responses should be taken into account. One can employ either one of two approaches: error correction and error tolerance. Responses $\widetilde{\mathbf{r}}$ typically have a $1-15\%$ error rate, considering their enrolled versions $\mathbf{r}$ as a reference, although it largely depends on the IC's environment. The lowest noise levels apply to laboratory settings where the environment is ultra-stable. Higher noise levels apply to practical in-the-field settings. Market products are typically supposed to function in a range of temperatures, among other specifications. For completeness, we mention the noisiness to be bit-specific: some bits are noisy, others are stable. Or more precisely, there is a continuous spectrum of bit error rates [18].

3.3 Counteracting Strong PUF Modeling Attacks (#3)

Strong PUFs are too fragile for unprotected exposure, as demonstrated by a history of machine learning attacks [19, 25, 26]. So far, no PUF architecture can claim to be practical, well-validated and robust against modeling. Or stated otherwise: no architecture satisfies the original strong PUF definition given in [5], as has been observed by others, e.g. [17]. With strong cryptographic primitives such as a hash function, one can fully mitigate this threat. Several proposals opt for more lightweight logic (XOR, PRNG, TRNG, etc.), although this might offer partial protection only, as becomes clear later-on.

An unprotected strong PUF able to resist modeling would be a real breakthrough. Unfortunately, two fundamental issues undermine the optimism. First, strong PUFs extract their enormous amount of bits from a limited IC area only, hereby using a limited number of circuit elements. The delay model of the arbiter PUF in Appendix A.2 of [1] provides an example. A highly correlated structure is the unavoidable consequence. Machine learning algorithms exploit these correlations in a 'blind' implicit manner. The modeling resistance is usually quantified by the minimum size of the training set as well as the algorithm runtime. More insightful explicit quantification has been initiated in [19]. Via simulations, one characterized the probability $P(r_u = r_v) = f(\mathbf{c}_u, \mathbf{c}_v)$ for the 'averaged' arbiter PUF. CRPs with $|f - 1/2| > 0$ are correlated. As a result of independent interest, we were the first to validate this work with an analytical model, as detailed in Appendix A.4 in [19]

As a second issue, the more entangled and diffusing the structure of a PUF, the more robust against modeling, but the less reproducible the responses as they accumulate more contributions from local noise sources. A popular modeling countermeasure is the replication of a PUF circuit, hereby XORing the output bits. This has been described for the arbiter PUF in Appendix A.3 in [1]. Correlations are considerably reduced as detailed in Appendix A.4 in [1]. Unfortunately, XORing amplifies the noisiness, posing a practical limit on the modeling resistance.

3.4 Strong PUF Response Space Expansion (#4)

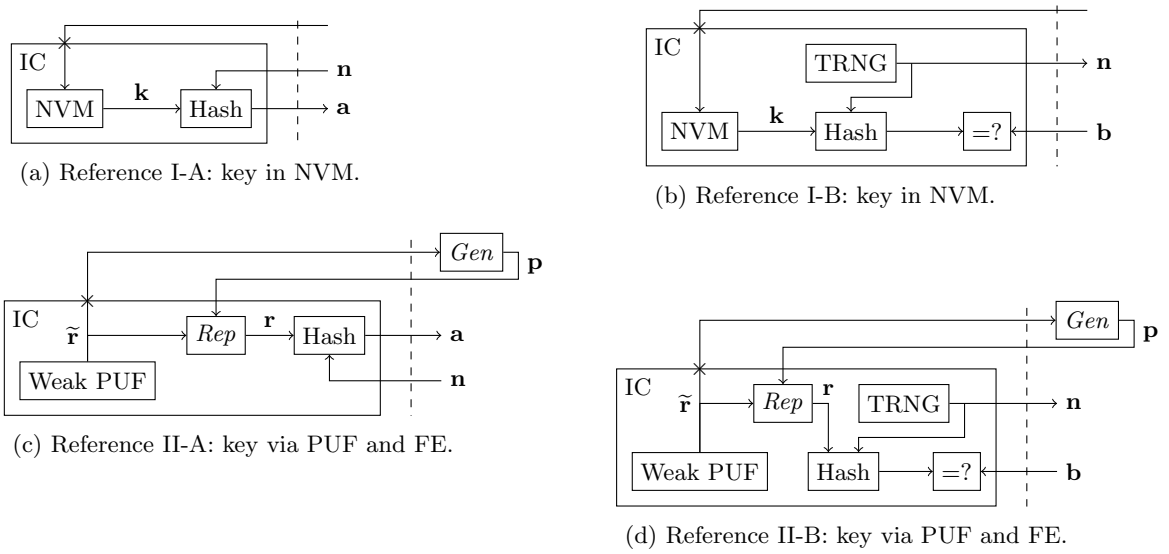
To counteract brute-force attacks and random guessing, all strong PUF protocols in this work require the challenge $\mathbf{c}$ and response $\mathbf{r}$ to be of sufficient length, e.g. $m = n = 128$. Unfortunately, strong PUFs provide a small response space only, often $n = 1$. Replicating the PUF circuit is a simple but unfortunately very expensive solution. The lightweight approach is to evaluate a list of n challenges, hereby concatenating the response bits. Several proposals suggest the use of a particular method to generate such a list. The server could send a list of true random numbers, requiring no additional IC logic, but resulting in a large communication overhead [6]. A small pseudorandom number generator (PRNG), such as a linear feedback shift register (LFSR), is often employed [2, 6, 14, 20, 24, 31, 33]. Challenge $\mathbf{c}$ is then used as a seed value: $\widetilde{\mathbf{r}} \leftarrow PUF(PRNG(\mathbf{c}))$. In [22], a repeated permutation is employed as PRNG. A variety of counter-based solutions can be applied as well [12]. We make abstraction of the response expansion method in the remainder of this work, except when there is a related security issue.

3.5 Efficiency (#5)

Efficiency is part of the PUF lightweight premise. We define our own PUF-based authentication protocol to establish a baseline. References II-A and II-B provide token and server authenticity respectively, as represented by Figures 1(c) and 1(d) respectively. They could again be combined in order to provide mutual authentication. Essentially, we employ a weak PUF to generate a secret key. Logic for generating challenges might be as simple as reading out the whole cell array. In principle, one could opt for a strong PUF with challenge generator as well. The response noisiness and non-uniformity issues are resolved with a FE. *Gen* is executed only once during enrollment. Public helper data $\mathbf{p}$ is stored by the server, or alternatively at the token side in insecure (off-chip) NVM. One could generate a key as $\mathbf{k} \leftarrow Hash(\mathbf{r})$. We perform an optimization by merging this step with the authentication hash: $\mathbf{a} \leftarrow Hash(\mathbf{r}, \mathbf{n})$.

3.6 Easy-to-instantiate (#6)

Ideally, one should not make exacting assumptions about the PUF, so that the protocol can be instantiated easily. There is considerable advantage in the design of generic PUF-independent protocols. First, efficiency



(a) Reference I-A: key in NVM.

(b) Reference I-B: key in NVM.

(c) Reference II-A: key via PUF and FE.

(d) Reference II-B: key via PUF and FE.

Fig. 1. Token representation for the references. The following IC logic is not drawn: intermediary registers (volatile) and control. A dashed line represent the interface with the server. One-time interfaces destructed after enrollment are marked by the symbol $\times$.

and performance characteristics differ for every PUF, in terms of speed, area, power, noisiness, etc. Flexibility of choice allows for better optimization with respect to a given set of constraints. Second, a recent string of physical attacks [7, 21, 27, 30] promotes to envision PUFs as easy-to-replace building blocks. Reference II embodies this vision as it functions with quasi every weak or strong PUF. The proposals under review are more specific: they all require a strong PUF. Furthermore, additional constraints are imposed, most frequently with respect to the modeling robustness.

3.7 Resistance against Protocol Attacks (#7)

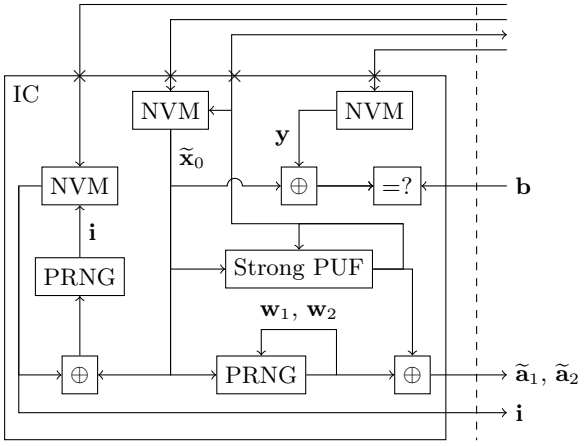
Resistance against conventional protocol attacks should be provided. We hereby assume the PUF to be an ideal random oracle, in order to distinguish from previous requirements. The primary threat is token and/or server impersonation. Furthermore, denial-of-service should be considered. This in addition to the validation of specific claims such as token privacy.

3.8 Identification Prior to Authentication (#8)

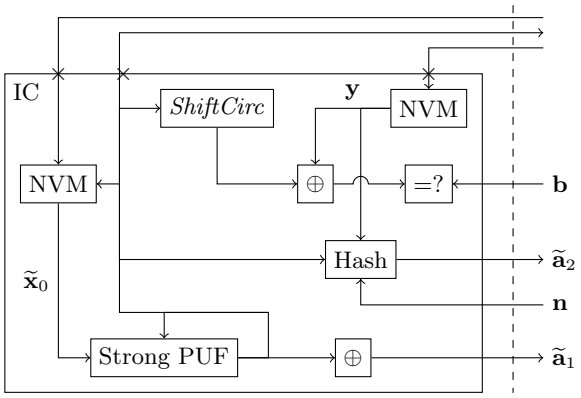
We distinguish between authentication and identification. The former comprehends the secure verification of an identity while the latter is limited to an unverified claim. Most protocols in this work provide token authentication and therefore also token identification. Unfortunately, identification prior to authentication is more practical. Otherwise, the server would have to iterate over all enrolled tokens to establish a match. There is a simple solution if token privacy is of no concern. Each token can store an identifier in insecure NVM [31]. CMOS-compatible fuses could be used to limit the manufacturing cost. Alternatively, one could opt for a PUF-generated identifier as well. Matching noisy identifiers is a straightforward operation and more complicated algorithms might even obtain sublinear complexity. We will not further comment on the absence of an identification step, except for the proposals which claim token privacy.

3.9 On the Mutual Authentication Order (#9)

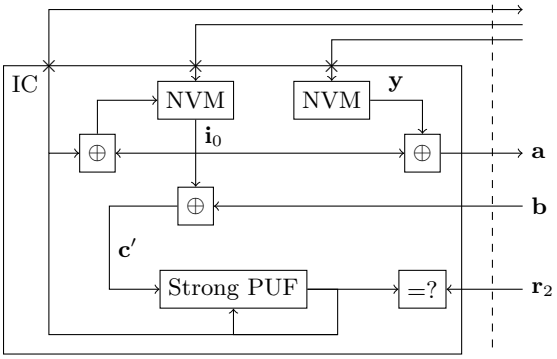
An attacker's potential to freely query tokens is a major concern. Therefore, it is a good practice to establish server authenticity first, in the case of mutual authentication [8]. The corresponding reduction in attack surface inherently benefits the security and privacy objectives. This might even be the reason to provide



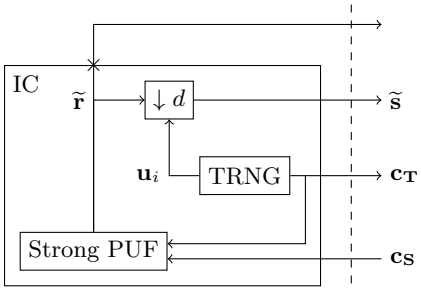
(a) Kulseng et al. [15].



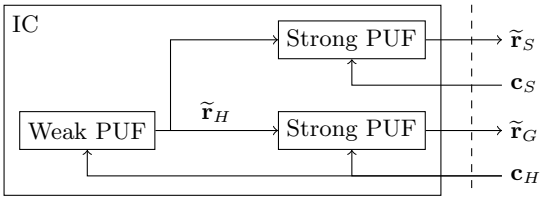
(b) Jin et al. [10].



(c) Xu et al. [32].



(d) Noise bifurcation [33].



(e) System of PUFs [14].

Fig. 2. Token representation for all strong PUF protocols. The following IC logic is not drawn: expansion of the strong PUF responses, intermediary registers (volatile) and control. A dashed line represent the interface with the server. One-time interfaces destructed after enrollment are marked by the symbol $\times$.

mutual authentication in the first place, rather than token authenticity only. Although several protocols employ the opposite order, we do not further discuss, as it comprehends a guideline rather that a stringent requirement.

4 Protocol Analysis

We describe and analyze all strong PUF protocols in chronological order. Sections 4.1 to 4.5 can be read in arbitrary order, although we recommend to get acquainted with the basic protocol in [1] first.

4.1 Kulseng et al. (March 2010)

The proposal of Kulseng et al. [15] provides mutual authentication. Figures 2(a) and 3 represent the protocol. Token identifier $\mathbf{i}$ is updated after every authentication, for improved privacy: tracking is only possible in between protocol executions. An attacker can desynchronize token and server by blocking the last message: only the former updates its state then. Therefore, recovery logic is foreseen to prevent denial-of-service from occurring. One suggest the use of an arbiter PUF, without acknowledging the need to expand the response space.

NVM Undermines PUF Benefit (#1) The need for secure NVM undermines the advantages of PUFs. Either read or write access would enable an attacker to break the system. Also, low-cost manufacturing is impeded, due to $\widetilde{\mathbf{x}}$ in particular as it requires reprogrammable NVM.

Non-Functional due to PUF Noisiness (#2) PUF responses are not acknowledged to be noisy, making the protocol non-functional. There will be a mismatch $\widetilde{\mathbf{x}}_1 = \mathbf{x}'_1 \oplus \mathbf{e}$, with $HW(\mathbf{e})$ small, leading to token rejection. The use of $\mathbf{x}_2$ is even more problematic: the feedback loop over the PUF leads to error amplification. We make abstraction of the noisiness issue in the remainder of our analysis.

Server Impersonation (#7) The desynchronization recovery logic allows for a simple replay attack. One can impersonate the server by resending the last $\mathbf{b}$.

Denial-of-Service (#7) As observed in [11], there is a simple denial-of-service attack. Interfering with a genuine protocol run and modifying $\widetilde{\mathbf{a}}_2$ to an arbitrarily chosen value is sufficient to desynchronize token and server.

Token/Server Impersonation and Privacy Breach (#7) The PRNG is instantiated with a LFSR. In [11], one describes an attack which leads to full system disclosure. Given an n-bit output sequence of an n-bit LFSR, one can easily retrieve the initial state. This principle is applied to two consecutive identifiers: $\mathbf{i}^{(1)}$ and $\mathbf{i}^{(2)}$. This allows for the recovery of $\widetilde{\mathbf{x}}_0^{(1)}$ and subsequently all other secret variables. Our analysis revealed an alternative PRNG exploitation attack, leading to full system disclosure as well. One can recover $\mathbf{w}_1^{(1)} = \mathbf{i}^{(2)} \oplus \mathit{LFSR}(\mathbf{i}^{(1)})$ and subsequently also other secret variables.

4.2 Jin et al. (April 2012)

The proposal of Jin et al. [10] seems to be inspired by [15]. The protocol is represented by Figures 2(b) and 4. The function *ShiftCirc* rotates its input over half the length. The authors claim resistance against physical attacks recovering the secret state $\widetilde{\mathbf{x}}_0$: future authentications are not threatened. The protocol is also claimed to be resistant against tracking. An attacker can desynchronize token and server by blocking the last message: only the latter updates its state then. Therefore, recovery logic is foreseen to prevent denial-of-service from occurring. The authors propose to instantiate the protocol with a variant of the arbiter PUF, without acknowledging the need to expand the response space.

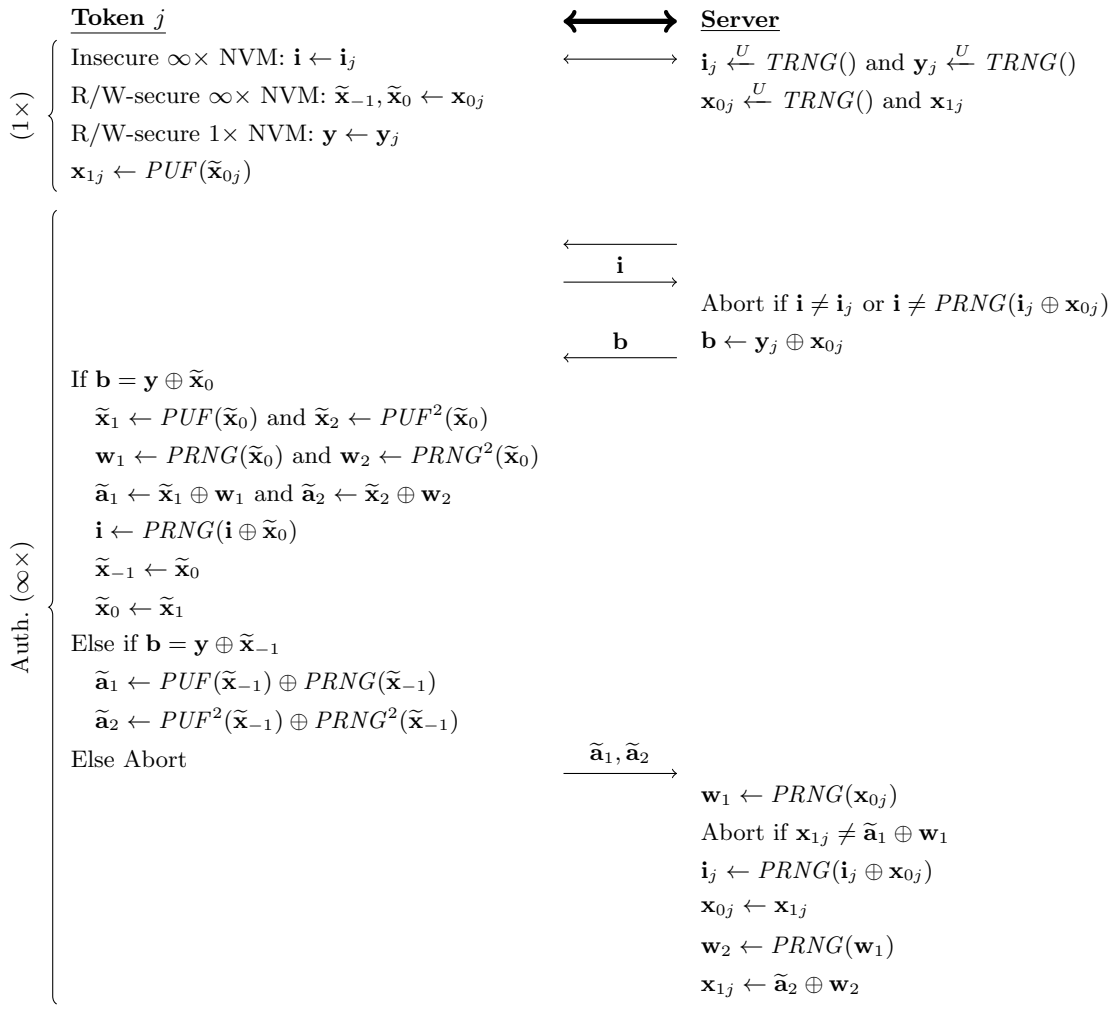


Fig. 3. Authentication protocol of Kulseng et al. There is contradictive information regarding the update of $\mathbf{i}$: either $\mathbf{x_0}$ or $\mathbf{x_1}$ may be used. Although it leads to an equally insecure system, we align our interpretation with the cryptanalysis in [11] and opt for $\mathbf{x_0}$.

NVM Issues (#1) The use of reprogrammable NVM undermines the low-cost advantage of PUF technology. The authors claim to be robust against key leakage via $\mathbf{x}'$, although they do not stress that this protection does not apply to $\mathbf{y}'$. The latter which is prone to physical attacks as well.

Non-Functional due to PUF Noisiness (#2) PUF responses are not acknowledged to be noisy, making the protocol non-functional. There will be a mismatch $\widetilde{\mathbf{x}}_1 = \mathbf{x}'_1 \oplus \mathbf{e}$, with $HW(\mathbf{e})$ small, leading to token rejection. The use of $\mathbf{x}_2$ is even more problematic: the feedback loop over the PUF leads to error amplification. We make abstraction of the noisiness issue in the remainder of our analysis.

Server Impersonation (#7) Eavesdropping on the last protocol execution allows for server impersonation. First one sends an arbitrary nonce $\mathbf{n}^{(2)}$ and subsequently one responds with $\mathbf{b}^{(2)} \leftarrow \mathbf{b}^{(1)} \oplus \mathit{ShiftCirc}(\widetilde{\mathbf{a}}_1^{(2)})$.

Denial-of-Service (#7) The proposal instantiates the hash computation as follows: $\widetilde{\mathbf{a}}_2 \leftarrow \mathit{Hash}(\mathbf{y}, \widetilde{\mathbf{x}}_2, \mathbf{n}) = \mathit{Hash}(\mathbf{y} \oplus \widetilde{\mathbf{x}}_2 \oplus \mathbf{n})$. This allows for desynchronization attacks: we describe two variations. With the first variant, one injects the following messages during a genuine protocol run: $\mathbf{n} \leftarrow \mathbf{n} \oplus \mathbf{e}$ and $\widetilde{\mathbf{a}}_1 \leftarrow \widetilde{\mathbf{a}}_1 \oplus \mathbf{e}$, with an arbitrary $\mathbf{e} \neq \mathbf{0}$. The server accepts and corrupts its state. With the second variant, one eavesdrops on

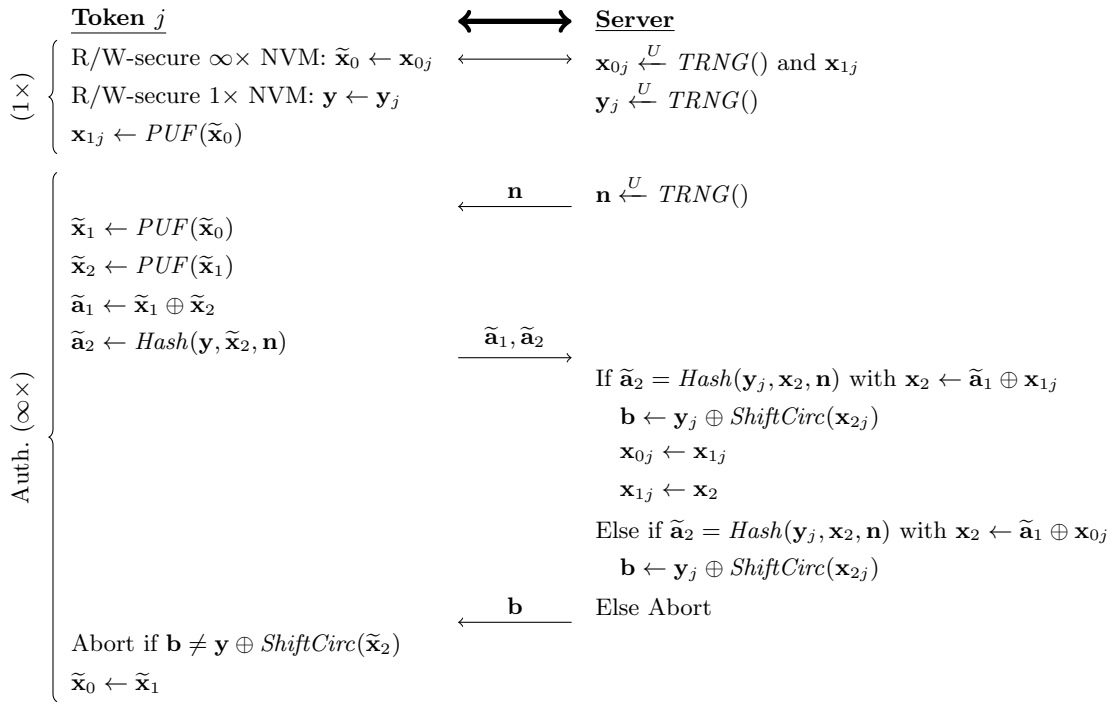


Fig. 4. Authentication protocol of Jin et al.

a genuine protocol run and obtains $\mathbf{n}^{(1)}$ and $\widetilde{\mathbf{a}}_2^{(1)}$. Subsequently, one replies with the following messages in the next protocol run: $\widetilde{\mathbf{a}}_2^{(2)} \leftarrow \widetilde{\mathbf{a}}_2^{(1)}$ and $\widetilde{\mathbf{a}}_1^{(2)} \leftarrow \mathbf{n}^{(1)} \oplus \mathbf{n}^{(2)}$. Again, the server accepts and corrupts its state. Furthermore, these attacks are in conflict with generally accepted privacy definitions. A single token which is persistently refused can easily be distinguished from its neighbors.

Token Impersonation via Leakage of $\widetilde{\mathbf{x}}_0$ (#7,#1) We argue the leakage claim to be incorrect. It allows for unlimited token impersonation, meanwhile also incapacitating the genuine token. An attacker eavesdrops on a genuine protocol run and obtains $\widetilde{\mathbf{a}}_1^{(1)}$. Subsequently, $\widetilde{\mathbf{x}}_0^{(2)}$ is obtained via leakage. Thereafter, $\widetilde{\mathbf{x}}_1^{(2)} \leftarrow \widetilde{\mathbf{a}}_1^{(1)} \oplus \widetilde{\mathbf{x}}_0^{(2)}$ can be computed. One can choose an arbitrary value for $\widetilde{\mathbf{x}}_2^{(2)}$. The attacker can now reply to the server nonce $\mathbf{n}^{(2)}$ and construct messages $\widetilde{\mathbf{a}}_1^{(2)}$ and $\widetilde{\mathbf{a}}_2^{(2)}$. The impersonation extends to an unlimited number of authentications: the PUF operation can be replaced with an arbitrary function.

Privacy without Identification (#8) The proposal does not scale in the number of tags. One cannot introduce a public identifier as this would oppose the privacy claim.

4.3 Xu et al. (September 2012)

The protocol of Xu et al. [32] is represented by Figures 2(c) and 5. There is a system key $\mathbf{y}$ which is shared by the server and all tokens. Token identifier $\mathbf{i}$ is updated after every authentication, for improved privacy: tracking is only possible in between protocol executions. An attacker can desynchronize token and server by blocking the last message: only the latter updates its state then. Therefore, recovery logic is foreseen to prevent denial-of-service from occurring. The authors do not acknowledge the need to expand the response space of the strong PUF.

Secure NVM (#1) The need for secure reprogrammable NVM undermines the PUF advantages: low-cost manufacturing and improved physical security. Furthermore, all eggs are put in the same basket by having a system key $\mathbf{y}$. If a single token is compromised, one can easily obtain the identifier of every token: $\mathbf{i}_0 \leftarrow \mathbf{a} \oplus \mathbf{y}$.

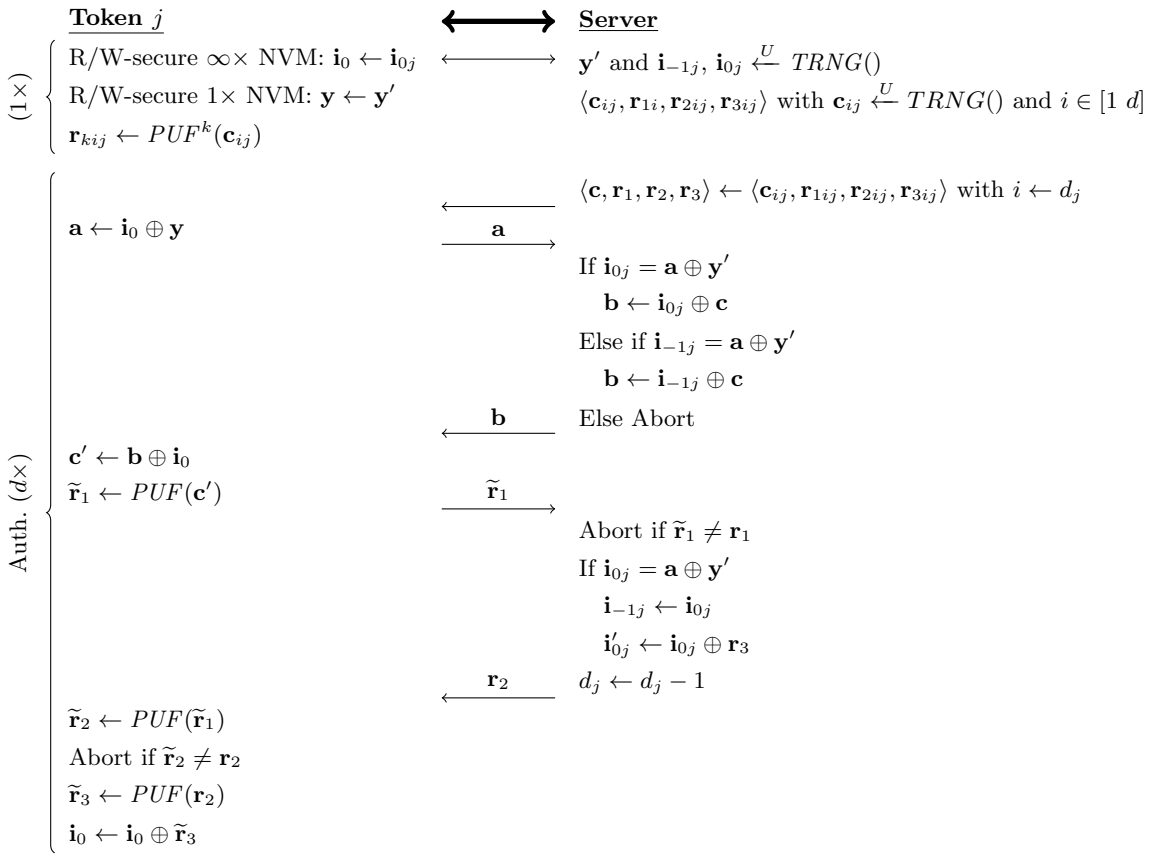


Fig. 5. Authentication protocol of Xu et al. There was only an incomplete and informal description of the desynchronization recovery logic. We filled in the blanks to the best of our insights and exclude this part from cryptanalysis.

Non-Functional due to PUF Noisiness (#2) PUF responses are not acknowledged to be noisy, making the protocol non-functional. There will be a mismatch $\widetilde{\mathbf{r}}_1 = \mathbf{r}_1 \oplus \mathbf{e}$, with $HW(\mathbf{e})$ small, leading to token rejection. The use of $\mathbf{r}_2$ and $\mathbf{r}_3$ is even more problematic: the feedback loop over the PUF leads to error amplification. We make abstraction of the noisiness issue in the remainder of our analysis.

Modeling Attacks (#3) The proposal does not acknowledge that PUFs are prone to modeling and offers no protection either. Most notably, the CRP $\langle \mathbf{r}_1, \mathbf{r}_2 \rangle$ is sent in the clear. Furthermore, one can obtain additional CRPs $\langle \mathbf{r}_2, \mathbf{r}_3 \rangle$, as $\mathbf{r}_3^{(1)} \leftarrow \mathbf{a}^{(1)} \oplus \mathbf{a}^{(2)}$. A practical limit on the number of authentications d should hence be imposed. If system key $\mathbf{y}$ would be compromised, one can freely query CRPs $\langle \mathbf{c}, \mathbf{r}_1 \rangle$.

Server Impersonation (#7) Eavesdropping on a single protocol run allows for unlimited server impersonation. An attacker gets authenticated with $\mathbf{b}^{(2)} \leftarrow \mathbf{b}^{(1)} \oplus \mathbf{a}^{(1)} \oplus \mathbf{a}^{(2)}$ and $\mathbf{r}_2^{(2)} \leftarrow \mathbf{r}_2^{(1)}$.

4.4 Noise Bifurcation (May 2014)

The noise bifurcation proposal [33] can be understood as a minor variation on the slender PUF protocol, as also reflected by Figure 2(d). There are three countermeasures against modeling, while avoiding the need for cryptographic primitives. First, the protocol requires a strong PUF with a high resistance. A model is constructed during enrollment via auxiliary one-time interfaces. The authors opt for arbiter XOR PUFs, as detailed in Appendix A.3 in [1]. Second, the exposure of $\widetilde{\mathbf{r}}$ is limited to a randomly decimated version $\widetilde{\mathbf{s}}$, hereby obfuscating the CRP link. The response string is partitioned into segments of d bits. For every segment, one generates a random number in order to discard all-but-one bits. This effect is referred to as learning 'noise'

for an attacker, not to be confused with physical noise. The server selects only the segments for which the bits are either all zero or all one, according to its PUF model. The method requires a pre-expansion of the PUF response with a factor $d \cdot 2^{d-1}$, posing a practical limit on the obfuscation. Third, server and token both contribute to the challenge via their respective nonces $\mathbf{c}_S$ and $\mathbf{c}_T$, counteracting chosen-challenge attacks. Figure 6 represents the protocol.

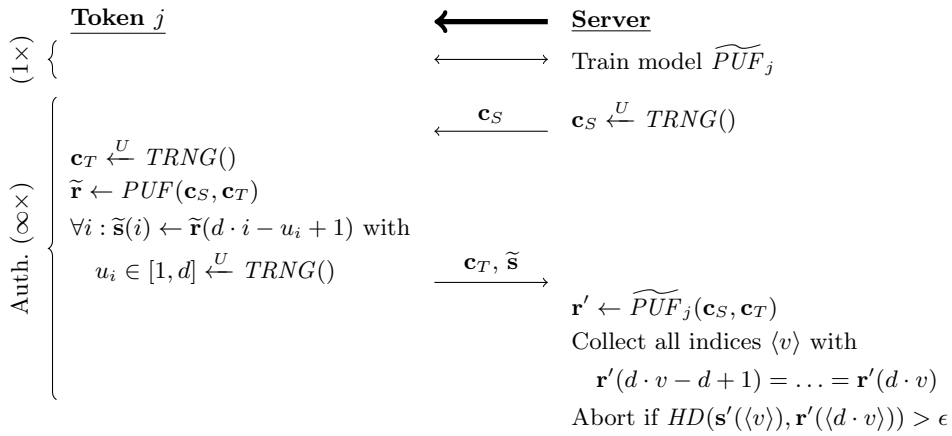


Fig. 6. Noise bifurcation protocol.

Exacting PUF Requirements (#6) The PUF requirements are rather exacting and partly opposing. On one hand, the PUF should be easy-to model, requiring a highly correlated structure. On the other hand, an attacker might exploit correlations either explicitly or implicitly. The proposed PUF architecture seems to offer this delicate balance. XORing reduces an attacker's potential although one-time interfaces allow for a bypass. During enrollment, the arbiter chains can be modeled separately. However, it all comes at a price: XORing amplifies noisiness, posing a practical limit on the number of chains. Specialized machine learning algorithms might therefore be successful. In the article, one was able to defeat the protocol using simulated arbiter XOR PUFs with up to four chains.

PRNG Unspecified (#7,#4) The protocol employs a PRNG to expand the PUF response space: $\widetilde{\mathbf{r}} \leftarrow PUF(PRNG(\mathbf{c}_S, \mathbf{c}_T))$. The authors suggest an LFSR-based design, without committing to a specific implementation. Therefore, we are not enable to properly evaluate the security. Remember that the reverse FE and slender PUF proposals were both found to be insecure due to their LFSR-based PRNG.

4.5 System of PUFs (October 2014)

The system of PUFs proposal [14] consists of three PUFs, as shown in Figure 2(e). They are referred to as hidden, guard and secure PUF. The hidden PUF is assumed not to be bothered by noise, as its response propagates to both of its neighbours. The secure PUF is assumed to be robust against modeling. Figure 7 represents the protocol. There is a two-level authentication. The first level, consisting of hidden PUF and guard PUF, is acknowledged to be insecure. Server and attacker face the exact same modeling burden here, as the enrollment is not aided by one-time interfaces. System security relies on the second level, the secure PUF. One claims that the protocol provides breach recognition and recovery. An attacker that modeled the first level, cannot provide the correct response for the second level. This would then be detected by the server, triggering a non-further specified recovery procedure. One suggests to instantiate hidden, guard and secure PUF with a ring oscillator, arbiter and arbiter XOR PUF respectively. One also claims robustness against denial-of-service attacks, unlike the basic authentication protocol, given an attacker which aims to deplete the server database.

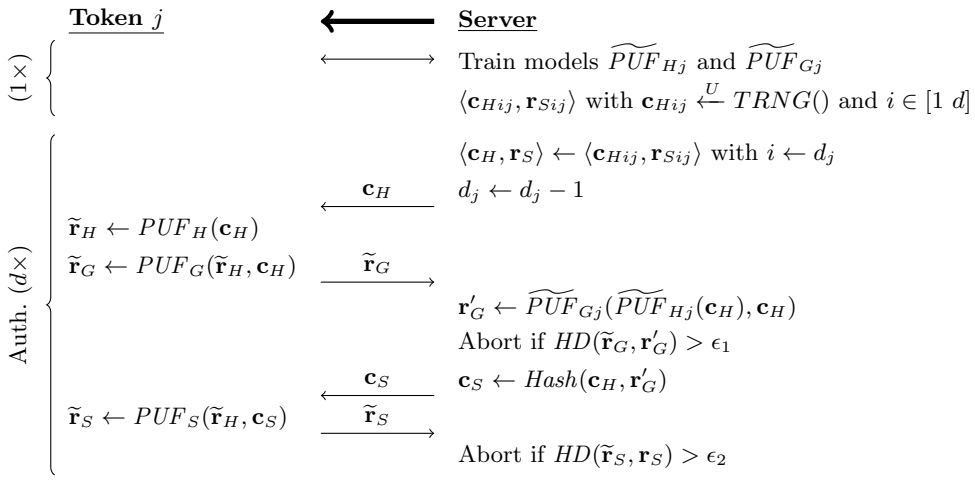


Fig. 7. System of PUFs.

PUF Noisiness Underestimated (#2) A single noisy bit of the hidden PUF is sufficient to result in an authentication failure. The authors rely on the ring oscillator PUF measurements in [29], offering outstanding error rates below 1%. However, they fail to mention that the reported error rate incorporates an error-correction scheme. With a raw PUF, the imposed requirement seems extremely hard to meet, especially under environmental perturbations.

Modeling Attacks (#3) Although the authors seem to suggest the opposite, there is no PUF which is robust against modeling. At least in practice, as noisiness provides e.g. an upper limit for the number of chains of an arbiter XOR PUF. There is hence no secure instantiation of the protocol.

No Need for the First Level (#5) Playing along with the assumption of a secure PUF, the first level would be superfluous. It does not improve system security and could be regarded as pure overhead. After its modeling, an attacker with physical access is free to query CRPs of the 'secure' PUF, which is no different from basic authentication. The authors derive argumentation from the breach recognition, but this concept is flawed as detailed hereafter. Also within the first level, one could question the need to protect the guard PUF with the hidden PUF, as it makes the enrollment more cumbersome. The authors argue that the prolonged modeling time eliminates many attack scenarios. However, it seems highly unlikely that minor differences in the physical access time would make a worthwhile difference. Furthermore, an attacker can focus its efforts on a single token, while the modeling burden of the server comprehends the complete set of tokens.

Flawed Breach Recognition Claim (#7) The potential for breach recognition is much lower than claimed. One assumes that an attacker attempts to defeat the server after modeling the first level, engaging in the protocol. However, playing along with the assumption of a secure PUF, this would be a useless effort. That's because random guessing of $\mathbf{r}_S$ should have a negligible probability of success. In practice, with insecure PUFs, an attacker might rather query a token until both levels are modeled. Furthermore, it is not clear how one would implement breach recovery without enabling a denial-of-service attack.

Flawed Server Depletion Claim (#7) The server depletion statements are unfair. It all depends on which party initiates the protocol, an aspect which has not been covered. In Figure 2 in [1], we represented the common sense version of the basic authentication protocol, with the server as sole initiator. The ability for a token to initiate would enable denial-of-service, but this is equally true for the system of PUFs.

		Reference II-A	**Reference II-B**	*Basic*	**Controlled**	*Öztürk et al.*	*Hammouri et al.*	*Kulseng et al.*	*Reconfiguration*	**Reverse FE**	**Reverse FE v2**	*Converse*	*Jin et al.*	**Slender**	*Xu et al.*	**Noise bifurcation**	*System of PUFs*
resources (#5)	Weak PUF	✓	✓	×	×	×	×	×	×	✓	×	×	×	×	×	✓	
	Strong PUF[1]	×	×	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	
	NVM[2]	×	×	×	×	✓	×	✓	✓	×	×	×	✓	×	×	×	×
	TRNG	×	✓	×	×	×	✓	×	×	×	✓	✓	×	✓	×	✓	×
	Gen	×	×	×	×	×	×	×	×	✓	✓	×	×	×	×	×	×
	Rep	✓	✓	×	×	×	×	×	×	×	×	✓	×	×	×	×	×
	Hash	✓	✓	×	✓	×	×	×	✓	✓	✓	✓	✓	×	×	×	×
	PRNG	×	×	×	×	×	×	✓	×	×	×	×	×	×	×	×	×
	$\oplus, =?$	×	✓	×	×	✓	✓	✓	×	✓	✓	✓	✓	×	✓	×	×
	$1\times$ interface	✓	✓	×	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	✓	×
claims	Server authenticity	×	✓	×	×	×	×	✓	×	✓	✓	✓	✓	×	✓	×	×
	Token authenticity	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	✓
	Token privacy	×	×	×	×	×	×	✓	✓	×	×	×	✓	×	✓	×	×
	# Authentications	∞	∞	d	d	∞	∞	∞	d	∞	∞	∞	∞	∞	d	∞	d
usability	PUF Benefit (#1)	✓	✓		✓					✓	✓			✓		✓	
	Noise Robust (#2)	✓	✓		✓	∼	×	×		✓	✓		×	✓	×	✓	∼
	Modeling Robust (#3)	✓	✓	×	✓				×	✓	✓			∼	∼	∼	×
	PUF Independency (#6)	✓	✓		∼					×	✓			×		×	
	Server authenticity (#7)		✓					×		∼	✓	×	×		×		
	Token authenticity (#7)	✓			✓			×		∼	✓			∼		✓	
	Token privacy (#7)							×									
	DoS prevention (#7)	✓	✓		✓	×		×	×	✓	✓		×	✓		✓	

[1] Including logic to expand the PUF response (#4).
[2] Not including identification (#8).

Table 1. For all protocols: token hardware (top segment), the authenticity and privacy claims (middle segment) and our condensed analysis (bottom segment). The symbol ✓ denotes 'yes'. The symbol × denotes 'no'. The symbol ∼ denotes the middle ground. An empty cell means 'non-applicable'. A grayed-out cell means 'irrelevant due to other issues'.

5 Overview and Discussion

Table 1 provides an overview of Section 4. We adopt the perspective of an interested **system provider, aiming to select a protocol**. Proposals which do not offer any robustness against both noise (#2) and modeling (#3) are discarded first. Subsequently, we discard proposals which are vulnerable to conventional protocol attacks (#7). Despite the exploitation of their response expansion method (#4), we maintain the slender PUF and original reverse FE proposals. This is due to the implementation-dependency: PRNG redesign might easily offer a fix for both. The seven remaining proposals are marked bold for further consideration. We do not object that the discarded proposals might contain worthwhile concepts.

The seven retained protocols can be split in two categories. The first category comprehends all forms of **PUF-based key generation**, including a strong cryptographic primitive to perform the authentication. This might provide excellent security, but it is unfortunately not so very lightweight (#5). Within this category, Reference II-A can be considered as the weak PUF variant of controlled PUFs. Similarly, there is also a weak PUF variant of the reverse FE protocol. Both weak PUF variants have the advantage that they are compatible with quasi every PUF (#6).

The second category comprehends **strong PUF obfuscation**. Although not equally secure, it improves the lightweight perspectives. However, both the slender PUF and noise bifurcation proposals rely on a TRNG to perform obfuscation. A physically secure TRNG is not easy to obtain in practice. Additional resources in the form of countermeasures might be required. Furthermore, this category only seems useful if the scope

is limited to just entity authentication. If there are other security request, e.g. message confidentiality and integrity, PUF-based key generation seems more appropriate.

6 Conclusion

Various protocols utilize a strong PUF to provide lightweight entity authentication. We described five proposals using a unified notation, hereby continuing our large-scale protocol overview of CHES 2014. Again, our analysis revealed numerous security and practicality issues. Most proposals aim to compensate the lack of cryptographic properties of the strong PUF. However, proper compensation seems to be in conflict with the lightweight objective. More fundamental physical research is required, aiming to create a truly strong PUF with great cryptographic properties. If not, we are inclined to recommend conventional PUF-based key generation as a more promising alternative. Furthermore, we are the first to explicitly list PUF protocol requirements. This might facilitate future protocol design.

7 Future Work

We plan to analyze more protocols in the near future. Therefore, please keep track of potential updates of this Eprint manuscript.

Acknowledgment

The authors greatly appreciate the support received. The European Commission through the ICT programme under contract FP7-ICT-2011-317930 HINT. The Research Council of KU Leuven: GOA TENSE (GOA/11/007), the Flemish Government through FWO G.0550.12N and the Hercules Foundation AKUL/11/19. The national major development program for fundamental research of China (973 Plan) under grant no. 2013CB338004. Jeroen Delvaux is funded by IWT-Flanders grant no. 121552.

References

1. J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede. Secure lightweight entity authentication with strong pufs: Mission impossible? In *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, pages 451–475, 2014.
2. S. Devadas, E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal. Design and implementation of puf-based "unclonable" rfid ics for anti-counterfeiting and security applications. pages 58–64. IEEE, 2008.
3. Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.
4. B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas. Silicon physical random functions. In V. Atluri, editor, *ACM Conference on Computer and Communications Security*, pages 148–160. ACM, 2002.
5. J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls. FPGA intrinsic pufs and their use for IP protection. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 63–80. Springer, 2007.
6. G. Hammouri, E. Öztürk, and B. Sunar. A tamper-proof and lightweight authentication scheme. *Pervasive and Mobile Computing*, 4(6):807–818, 2008.
7. C. Helfmeier, C. Boit, D. Nedospasov, and J. Seifert. Cloning physically unclonable functions. In *2013 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2013, Austin, TX, USA, June 2-3, 2013*, pages 1–6. IEEE, 2013.
8. J. Hermans, R. Peeters, and J. Fan. IBIHOP: Proper Privacy Preserving Mutual RFID Authentication. In *Workshop on RFID and IoT Security - RFIDSec Asia 2013*, Cryptology and Information Security, pages 45–56, Guangzhou, China, 2013. IOS PRESS.
9. D. E. Holcomb, W. P. Burleson, and K. Fu. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Trans. Computers*, 58(9):1198–1210, 2009.
10. Y. Jin, W. Xin, H. Sun, and Z. Chen. Puf-based rfid authentication protocol against secret key leakage. In Q. Z. Sheng, G. Wang, C. S. Jensen, and G. Xu, editors, *APWeb*, volume 7235 of *Lecture Notes in Computer Science*, pages 318–329. Springer, 2012.

11. S. Kardaş, M. Akgün, M. S. Kiraz, and H. Demirci. Cryptanalysis of lightweight mutual authentication and ownership transfer for rfid systems. In *Lightweight Security Privacy: Devices, Protocols and Applications (LightSec), 2011 Workshop on*, pages 20–25, March 2011.
12. S. Katzenbeisser, Ünal Koçabas, V. van der Leest, A. Sadeghi, G. J. Schrijen, H. Schröder, and C. Wachsmann. Recyclable pufs: Logically reconfigurable pufs. In B. Preneel and T. Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 374–389. Springer, 2011.
13. Ünal. Kocabaş, A. Peter, S. Katzenbeisser, and A. Sadeghi. Converse puf-based authentication. In S. Katzenbeisser, E. Weippl, L. J. Camp, M. Volkamer, M. K. Reiter, and X. Zhang, editors, *Trust and Trustworthy Computing - 5th International Conference, TRUST 2012, Vienna, Austria, June 13-15, 2012. Proceedings*, volume 7344 of *Lecture Notes in Computer Science*, pages 142–158. Springer, 2012.
14. S. T. C. Konigsmark, L. K. Hwang, D. Chen, and M. D. F. Wong. System-of-pufs: Multilevel security for embedded systems. In *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*, CODES '14, pages 27:1–27:10, New York, NY, USA, 2014. ACM.
15. L. Kulseng, Z. Yu, Y. Wei, and Y. Guan. Lightweight mutual authentication and ownership transfer for rfid systems. In *INFOCOM*, pages 251–255. IEEE, 2010.
16. J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *Symposium on VLSI Circuits*, pages 176–179. IEEE, 2004.
17. R. Maes. *Physically Unclonable Functions: Constructions, Properies and Applications*. PhD thesis, KU Leuven, 2012.
18. R. Maes. An accurate probabilistic reliability model for silicon pufs. In G. Bertoni and J. Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 73–89. Springer, 2013.
19. M. Majzoobi, F. Koushanfar, and M. Potkonjak. Testing techniques for hardware security. In D. Young and N. A. Touba, editors, *ITC*, pages 1–10. IEEE, 2008.
20. M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas. Slender puf protocol: A lightweight, robust, and secure authentication by substring matching. In *IEEE Symposium on Security and Privacy Workshops*, pages 33–44. IEEE Computer Society, 2012.
21. D. Merli, D. Schuster, F. Stumpf, and G. Sigl. Semi-invasive em attack on fpga ro pufs and countermeasures. In *Workshop on Embedded Systems Security*, 2011.
22. E. Öztürk, G. Hammouri, and B. Sunar. Towards robust low cost authentication for pervasive devices. In *PerCom*, pages 170–178. IEEE Computer Society, 2008.
23. R. S. Pappu. *Physical One-Way Functions*. PhD thesis, MIT, 2001.
24. Z. S. Paral and S. Devadas. Reliable and efficient puf-based key generation using pattern matching. In *HOST*, pages 128–133. IEEE Computer Society, 2011.
25. U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber. Modeling attacks on physical unclonable functions. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 237–249. ACM, 2010.
26. U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas. Puf modeling attacks on simulated and silicon data. *IEEE Transactions on Information Forensics and Security*, 8(11):1876–1891, 2013.
27. U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, M. Majzoobi, F. Koushanfar, and W. Burleson. Efficient power and timing side channels for physical unclonable functions. In *CHES*. IEEE, 2014.
28. S. P. Skorobogatov. Semi-invasive attacks - a new approach to hardware security analysis. Technical Report UCAM-CL-TR-630, University of Cambridge, Computer Laboratory, 2005.
29. G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *DAC*, pages 9–14. IEEE, 2007.
30. S. Tajik, E. Dietz, S. Frohmann, J. Seifert, D. Nedospasov, C. Helfmeier, C. Boit, and H. Dittrich. Physical characterization of arbiter pufs. In *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, pages 493–509, 2014.
31. A. Van Herrewege, S. Katzenbeisser, R. Maes, R. Peeters, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann. Reverse fuzzy extractors: Enabling lightweight mutual authentication for puf-enabled rfids. In A. D. Keromytis, editor, *Financial Cryptography*, volume 7397 of *Lecture Notes in Computer Science*, pages 374–389. Springer, 2012.
32. Y. Xu and Z. He. Design of a security protocol for low-cost rfid. In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2012 8th International Conference on*, pages 1–3, 2012.
33. M.-D. M. Yu, D. M'Raïhi, I. Verbauwhede, and S. Devadas. A noise bifurcation architecture for linear additive physical functions. In *HOST*, pages 124–129. IEEE, 2014.