A Survey on Lightweight Entity Authentication with Strong PUFs

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 this work, we review nineteen proposals in chronological order: from the original strong PUF proposal (2001) to the more complicated noise bifurcation and system of PUF proposals (2014). The assessment is aided by a unified notation and a transparent framework of PUF protocol requirements. The work comprehends a sequel on our prior CHES 2014 manuscript[1].

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: radio-frequency identification (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 [41] 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.

Numerous **PUF-based authentication protocols** have been proposed. Two categories of protocols naturally emerge, linked to functional discrepancies between so-called weak and strong PUFs. The weak PUF

[1] This work extends our prior CHES 2014 manuscript "Secure Lightweight Entity Authentication with Strong PUFs: Mission Impossible?" as follows. Most notably, eleven additional strong PUF protocols are included in the overview-analysis. Also, we improved the transparency of our analysis by explicitly listing protocol requirements. Finally, token privacy has been included in the analysis.

approach comprehends the use of PUFs as key generator, which is perfectly compatible with conventional authentication protocols. We focus on the strong PUF approach instead. The input-output behavior of the PUF is then an essential component of the protocol itself. This requires a special flavor of protocol design. We are the first to construct an extensive list of protocol requirements and apply it as an analysis tool to a rather complete list of strong PUF protocols. Numerous security and practicality issues appear. To such an extent, that we cannot support the use of most proposals in their current form. Our framework of protocol requirements is also intended as a checklist to improve the quality of future PUF-based protocols.

We consider the following **list of strong PUF protocols**, all described with a unified notation for ease of understanding: basic strong PUF authentication [34], controlled PUFs [8], Bolotnyy et al. [2], Öztürk et al. [33], Hammouri et al. [10], Kulseng et al. [23], Sadeghi et al. [39], logically reconfigurable PUFs [19], reverse fuzzy extractors [45], the converse protocol [20], Lee et al. I [25], Jin et al. [15], slender PUFs [30], Xu et al. [46], He et al. [11], Jung et al. [16], Lee et al. II [26], noise bifurcation [47] and system of PUFs [21].

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 initializes variables in NVM and/or obtains some information about the PUF to establish a shared secret. The destruction of one-time interfaces might permanently restrict direct NVM/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 manuscript is organized as follows. Section 2 introduces notation and preliminaries. Section 3 lists the requirements of a PUF-based authentication protocol. Section 4 introduces authentication via PUF-based key generation, to be considered as a reference. Section 5 describes and analyzes all strong PUF protocols. Our analysis is performed at protocol level, considering PUFs as a black box. Section 6 provides an overview of the protocol assessment. Section 7 concludes the work.

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 Embedded Non-Volatile Memory

We consider three categories of embedded non-volatile memory (NVM). They are listed with increasing flexibility in terms of programmability, as it determines which types of variables they can support. The manufacturing cost often translates to additional masks and processing steps with respect to a regular CMOS process. The first category is **hard-wired read-only** memory (ROM). Its contents are mask-defined and therefore shared by each token. Technologies are typically low-cost as the manufacturing process is not further complicated. The second category is **one-time programmable** (OTP) NVM. Technologies are based on either fuses or antifuses and are either foundry-specific or foundry-independent. Several IP providers offer foundry-independent OTP NVMs which do not further complicate the manufacturing process. This includes eMemory, Kilopass, Novocell Semiconductor, NSCore and Sidense. To a very limited extent and at a high area cost, reprogramming can be emulated via the partitioning of a large OTP NVM. Writing comprehends disabling and enabling the current and next partition respectively. The third category is true **multiple-time programmable** (MTP) NVM, supporting a very high number of write-cycles. Unfortunately, traditional technologies are expensive. EEPROM and its successor Flash require floating gate transistors, resulting in additional masks and processing steps. Battery-backed SRAM relies on CMOS-compatible SRAM, but batteries are considered to be very expensive. There are MTP NVM cell structures which do not further

complicate the manufacturing process, as provided by, e.g., eMemory. However, these initiatives should not be considered as a commodity yet. Furthermore, the storage density is typically lower than EEPROM/Flash.

2.3 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 [14] and the ring oscillator PUF[2] [42] 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 [6], 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 [24]. 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 [37]. For completeness, we note that the definition of strong PUFs has shifted over the years. The original more specific notion in [9] 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.

PUFs were initially praised for their resistance against **physical attacks**. Hereby often relying on the intuitive insight that invasion of the IC might damage the PUF and hence destroy the secret. This is in addition to the volatile nature of PUFs, posing limits on the attack time. However, a recent string of physical attacks [12, 31, 38, 43] reduces the optimism. Side-channel analysis is non-invasive and cannot destroy the secret. Even invasive techniques were demonstrated to be successful. The primary focus of these attacks is collecting CRPs. For weak PUFs, this directly corresponds with a characterization of the secret. For strong PUFs, an additional machine learning step might be required for the attack to be successful. Despite all the former, it needs to be said that there is hardly work on countermeasures. Just like NVM and algorithms without countermeasures are an easy target, the same might be true for PUFs.

2.4 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. Although a wide variety of reconstruction methods has been proposed [4], secure sketches [6, 7] are a particularly useful 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 [45].

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

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 $\mathbf{e}$
		$\mathbf{r} \leftarrow \widetilde{\mathbf{r}} \oplus \mathbf{e}$

A fuzzy extractor (FE) can be constructed out of a secure sketch. An additional hash function then produces a nearly uniform output: $\mathbf{k} \leftarrow \mathit{Hash}(\mathbf{r})$. By having more input than output bits, it compensates for the non-uniformity of $\mathbf{r}$ as well as the helper data leakage. Although no generic threats have been reported, we stress that other methods, used either as an alternative or an extension, are known to be vulnerable against helper data manipulation attacks. Therefore, it can be advisable to extend all of the former with a helper data integrity check [3].

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 3.10 describe a mixture of PUF-induced requirements and more conventional concerns. The list is tailored to the vast majority of PUF protocols, aiming to obtain token-server entity authentication, but can be mapped quite easily to other settings.

3.1 Complete Specification (#1)

A protocol should be specified in a complete unambiguous manner. Although this seems obvious, we observe that many proposals do not comply. Furthermore, the use of a particular PUF should be suggested, at least for protocols of which the security and/or functional behavior is PUF-dependent. As a side note, there is considerable advantage in having a graphical representation of a protocol, clearly detailing all computations and exchanged messages. This facilitates the analysis considerably. We observe that many proposals focus on a text-based description, with only a minimal graphical representation as support. In this work, we represent all protocols in a detailed graphical manner, hereby using a unified notation. For ease of understanding, we make abstraction of non-essential refinements regarding the two-party setting. E.g., multiple servers, a trusted issuer to aid the enrollment, a distinction between RFID readers and the back-end server, etc.

3.2 Leakage Resilience (#2)

The main advantage of PUF technology with respect to NVM should be preserved: improved physical security. Otherwise, one could equally well opt for traditional secret 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. Therefore, we argue that PUF protocols should be resilient against the leakage of their NVM contents. Although not equally critical, the extended case where an attacker also has NVM write-access would correspond to a stronger security claim. Also, we stress that protocol components other than NVM should be implemented in a physically secure manner. E.g., with an unprotected cryptographic algorithm, NVM might not be the weakest link anymore.

3.3 Able to Handle Noisiness (#3)

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 [28].

3.4 Counteracting Strong PUF Modeling Attacks (#4)

Strong PUFs are too fragile for unprotected exposure, as demonstrated by a history of machine learning attacks [24,29,36,37]. 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 [9], as has been observed by others, e.g., [27]. 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 [36] 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 [29]. 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 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 countermeasure against modeling is the replication of a strong PUF circuit, hereby XORing the output bits. As has been analyzed for arbiter PUFs, correlations are considerably reduced then. Unfortunately, XORing amplifies the noisiness, posing a practical limit on the modeling resistance.

3.5 Strong PUF Response Space Expansion (#5)

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 [19,20]. 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 [10]. A small pseudorandom number generator (PRNG), such as a linear feedback shift register (LFSR), is often employed [5, 10, 21, 30, 35, 45, 47]. Challenge $\mathbf{c}$ is then used as seed value: $\widetilde{\mathbf{r}} \leftarrow \mathit{SPUF}(\mathit{PRNG}(\mathbf{c}))$. In [33], a repeated permutation is employed for the same purpose. One might also be able to reuse cryptographic primitives, e.g., in [19], a challenge with an additional counter input is repeatedly hashed. We make abstraction of the response expansion method in the remainder of this work, except when there is a related security issue.

3.6 Low-Cost and Resource-Constrained (#6)

We evaluate the PUF lightweight premise. Low-cost manufacturing is mainly an issue for proposals which rely on MTP NVM. However, as practically every protocol building block requires a physically secure implementation, it might extend to a more general concern, largely depending on the selected countermeasures. Constraints in available resources are mainly an issue for proposals which rely on strong cryptographic primitives, such as a hash function.

3.7 Easy-to-instantiate (#7)

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 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, the recent string of physical attacks on PUFs promotes to envision them as easy-to-replace building blocks. We note that all protocols in this work require physical attacks on the PUF to be infeasible. PUF-based key generation is compatible 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.

3.8 Resistance against Protocol Attacks (#8)

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 (DoS) attacks should be considered. We limit the scope to protocol-based DoS attacks, as they can be executed remotely. This because mechanical DoS attacks are trivial to perform, given that one might acquire physical access to a token. Specific protocol claims such as token privacy should be validated as well. For protocols which claim NVM leakage resilience, all of the former should be reevaluated. It even makes sense to reevaluate DoS, as it could have a benefit to perform such an attack with delay, in a non-mechanical manner.

The assumed attacker capabilities should be realistic with respect to the intended application, as it its an occasional practice to mitigate realistic threats with unrealistic assumptions. Luckily, most proposals stick to a fully insecure channel between token and server as well as physical attacks on the token state. We stress the importance of assuming that a token and its PUF are still functional after performing the NVM leakage attack. If an attacker cannot interfere with a genuine protocol run anymore, the security claim would be very marginal. Note that server impersonation, token privacy and DoS are irrelevant for a broken token.

3.9 Scalability: Identification Prior to Authentication (#9)

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 OTP NVM [45]. 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.

3.10 On the Mutual Authentication Order (#10)

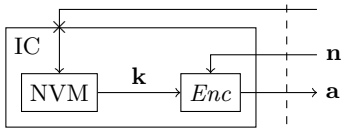
An attacker's potential to freely query tokens is a major concern. Therefore, it is a good practice to establish server authenticity first, in the case of mutual authentication [13]. The corresponding reduction in attack surface inherently benefits the security and privacy objectives. This might even be the reason to provide 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 Authentication via PUF-Based Key Generation

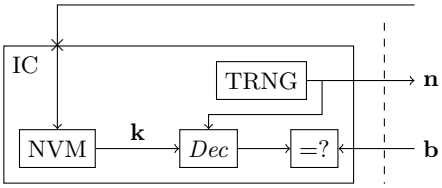
In principle, one could select a conventional authentication protocol from literature and replace key-storage in NVM with PUF-based key generation, as the latter is believed to be more physically secure. Although such concatenations are not the primary interest of this work, they establish a baseline in terms of security, efficiency and extensibility. A well-validated protocol combined with a secure mechanism for PUF-based key generation might provide excellent security guarantees. Efficiency might be a major concern though as strong cryptographic primitives are expected to be involved. Extensibility to other requirements is excellent and one might benefit from a wide body of literature. Guided by Figure 1, we now refine foregoing sketch on concatenated protocols.

4.1 Reference Protocols

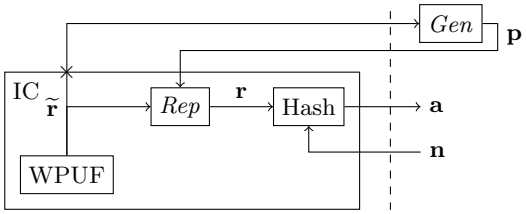
Reference I-A in Figure 1(a) relies on physically secure NVM, an assumption which might not always hold in practice. Figure 2 represent the corresponding protocol, which offers token authenticity only. Each token stores a unique secret key $\mathbf{k}$ in OTP NVM. A cryptographic primitive performs the authentication. We opt



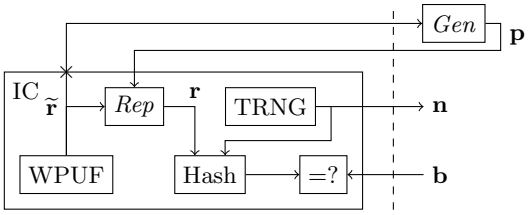
(a) Reference I-A: key in NVM.



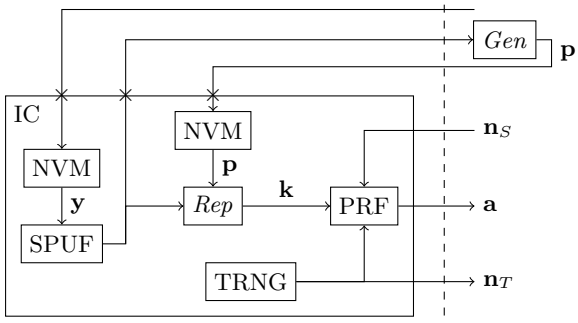
(b) Reference I-B: key in NVM.



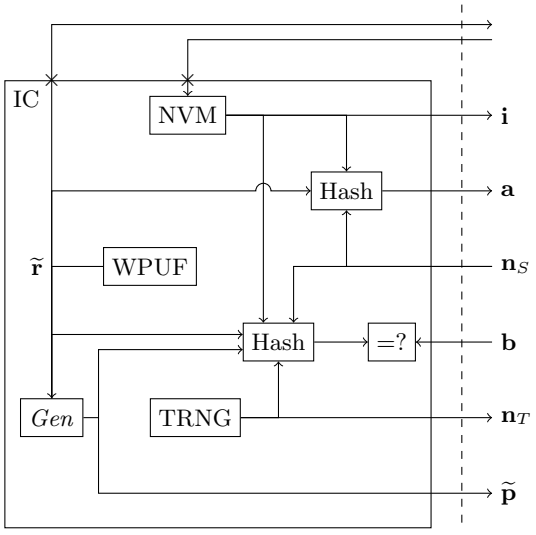
(c) Reference II-A: PUF-based key



(d) Reference II-B: PUF-based key.



(e) Sadeghi et al. [39].



(f) Modified reverse FE [27].

Fig. 1. Token representations for the key generation approach. 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 $\times$.

for a block cipher with encryption and decryption algorithms *Enc* and *Dec* respectively. E.g., the 128-bit version of the Advanced Encryption Standard (AES) might be used. Reference I-B, represented by Figure 1(b) and 3, provides server authenticity. Both mechanisms could be combined in order to provide mutual authentication.

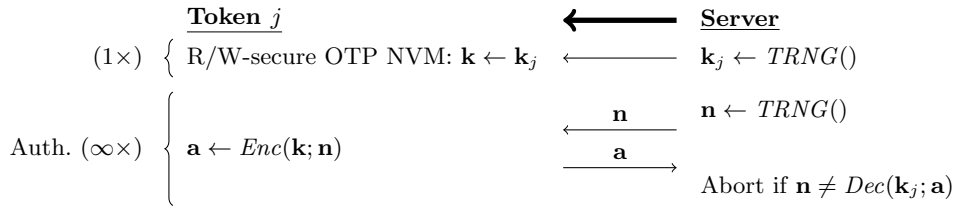


Fig. 2. Authentication with Reference I-A. The thick arrow points from verifier to prover.

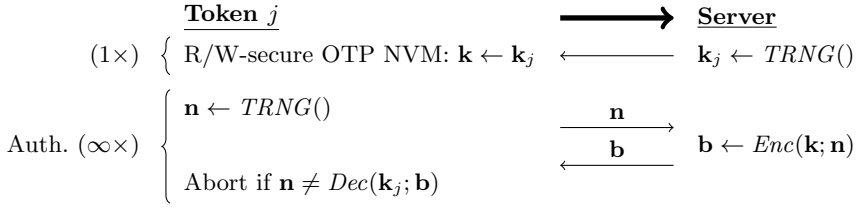


Fig. 3. Authentication with Reference I-B.

Reference II-A, represented by Figures 1(c) and 4, provides token authenticity. A weak PUF (*WPUF*) is sufficient and logic for generating challenges might be as simple as reading out the whole cell array. One could opt for a strong PUF (*SPUF*) with challenge generator as well. The response noisiness and non-uniformity issues are resolved with a fuzzy extractor. *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 OTP NVM. One could generate a key as $\mathbf{k} \leftarrow Hash(\mathbf{r})$. We perform an optimization by merging this step with the authentication. Reference II-B, represented by Figure 1(d) and 5, provides server authenticity. Again, both mechanisms could be combined in order to provide mutual authentication.

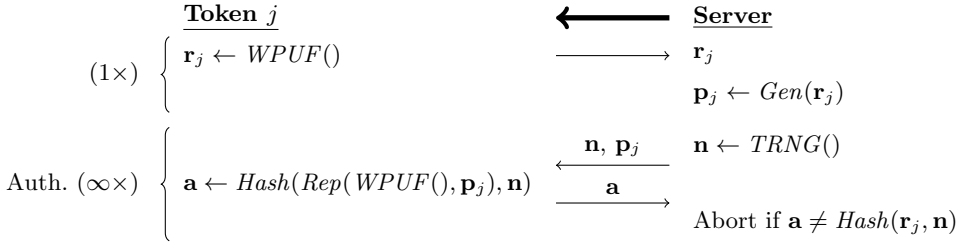


Fig. 4. Authentication with Reference II-A.

4.2 Existing Literature

Several proposals in literature aim to provide entity authentication via PUF-based key generation, as listed next. Tuyls et al. [44] use public key cryptography, mitigating the need for a shared secret between token and server, although it might not be so very lightweight. The proposal of Sadeghi et al. [39] is formulated as a strong PUF protocol and hence analyzed later-on, although it can be reduced quite trivially to a weak PUF protocol. The modified reverse fuzzy extractor proposal [27] comprehends a special case of secret key generation, as the key is adapted to the noise. There is the protocol of Bassil et al. [1], which has been badly

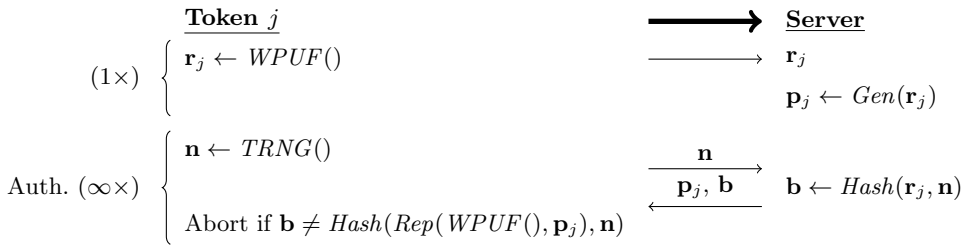


Fig. 5. Authentication with Reference II-B.

broken in [40]. The proposal of Kardas et al. [18] is formulated as a strong PUF protocol, although it can again be reduced quite trivially to a weak PUF protocol. However, it does not fully fit our scope, as it aims to defend against RFID reader compromise.

5 Strong PUF Protocol Analysis

We describe and analyze all strong PUF protocols in chronological order, guided by Figure 6. As some protocols contain similar design concepts, we also considered categorizing them. However, it does not fly all the way, so we opt for the history of development as main theme instead. Protocols already discussed in our prior CHES 2014 manuscript are not included here. Sections 5.1 to 5.11 can be read in arbitrary order, although we recommend to get acquainted with the basic protocol first.

5.1 Bolotnyy et al. (March 2008)

The protocol of Bolotnyy et al. [2] is represented by Figures 6(c) and 7. PUFs are assumed to be robust against modeling. The authors claim token privacy under the assumption of an eavesdropping attacker.

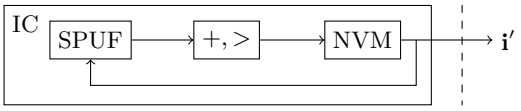
Incomplete Specification (#1) The authors do not acknowledge the need to expand the response space of the strong PUF.

NVM Undermines PUF Benefit (#2) The need for secure NVM undermines the main benefit of PUF technology: improved physical security. Leakage of the state allows for one-time token impersonation as well as a one-time privacy breach. This issue could be resolved by storing the preceding identifier rather than the currently valid identifier. With write-access, an attacker could reconstruct the server database.

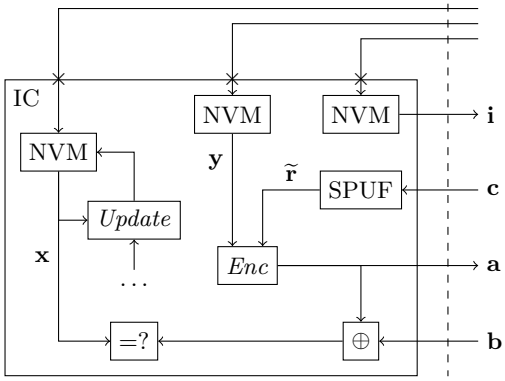
Non-Functional due to PUF Noisiness (#3) PUF noisiness has not been taken into account properly, making the protocol non-functional. The authors suggest majority voting as an error-correction mechanism, hereby evaluating the PUF multiple times to approach the most likely response value. Under the assumption of a uniform bit error rate, one could make the failure rate arbitrarily small as such. However, in practice, bit error rates are highly non-uniform, so a majority vote by itself is ineffective. Bits with error rates close to 50% remain problematic.

Modeling Attacks (#4) It is unrealistic to assume that PUFs are robust against modeling, which further limits the practical value of the protocol. There is no secure instantiation due to the lack of an appropriate strong PUF.

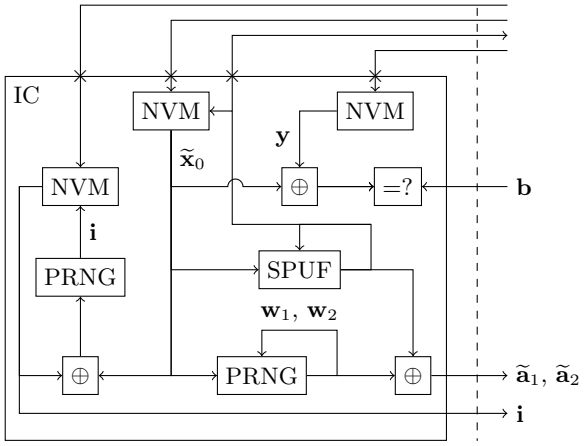
Overly Restricted Attacker Capabilities, Denial-of-Service (#8) As acknowledged in the article, the assumption of an eavesdropping attacker mitigates a denial-of-service threat. However, we argue that realistic threats should not be mitigated with unrealistic assumptions. Sending an authentication request to a token is sufficient to cause desynchronization. Either blocking or modifying a token response has a similar effect.



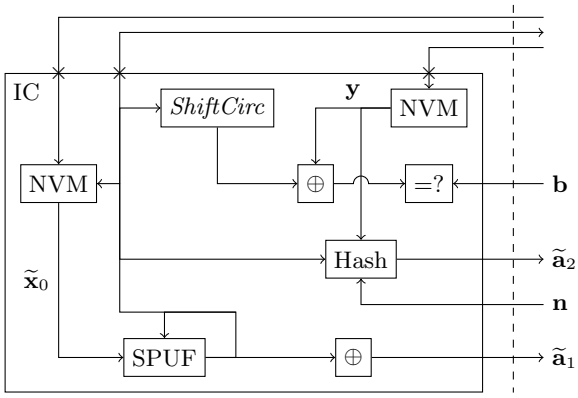
(c) Bolotnyy et al. [2].



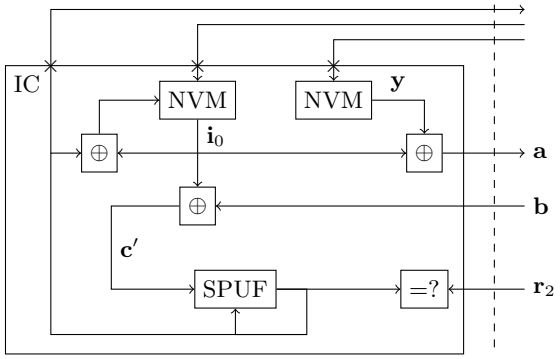
(j) Lee et al. I [25].



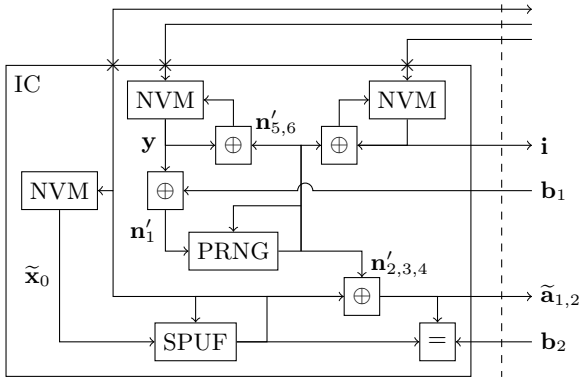
(f) Kulseng et al. [23].



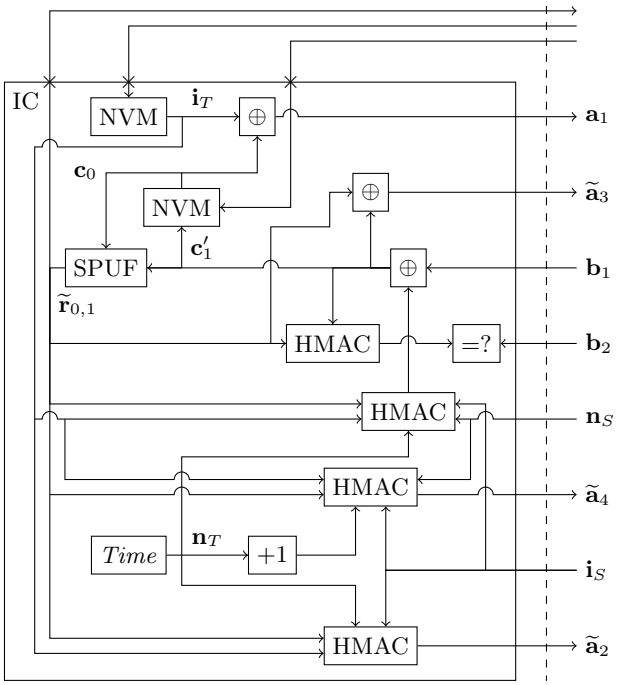
(k) Jin et al. [15].



(m) Xu et al. [46].

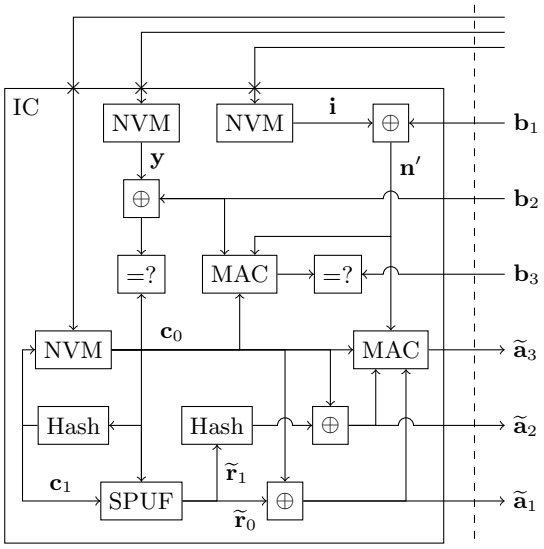


(n) He et al. [11].

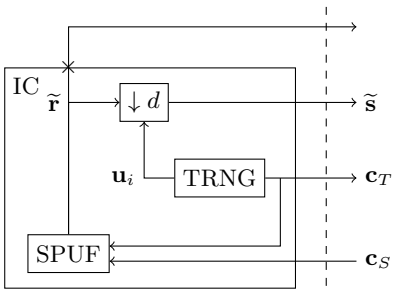


(o) Jung et al. [16].

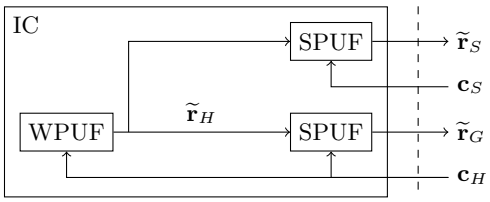
Fig. 6. Token representation for all strong PUF protocols. The following IC logic is not drawn: expansion of the strong PUF responses, intermediary registers (volatile) and control.



(p) Lee et al. II [26].



(q) Noise bifurcation [47].



(r) System of PUFs [21].

Fig. 6. Token representation for all strong PUF protocols (continued).

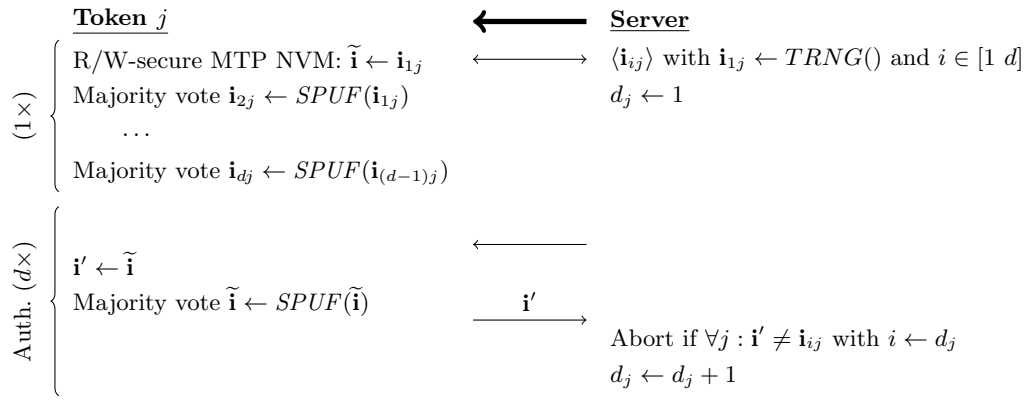


Fig. 7. Authentication protocol of Bolotnyy et al.

5.2 Kulseng et al. (March 2010)

The proposal of Kulseng et al. [23] provides mutual authentication. Figures 6(f) and 8 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. The authors suggest the use of an arbiter PUF.

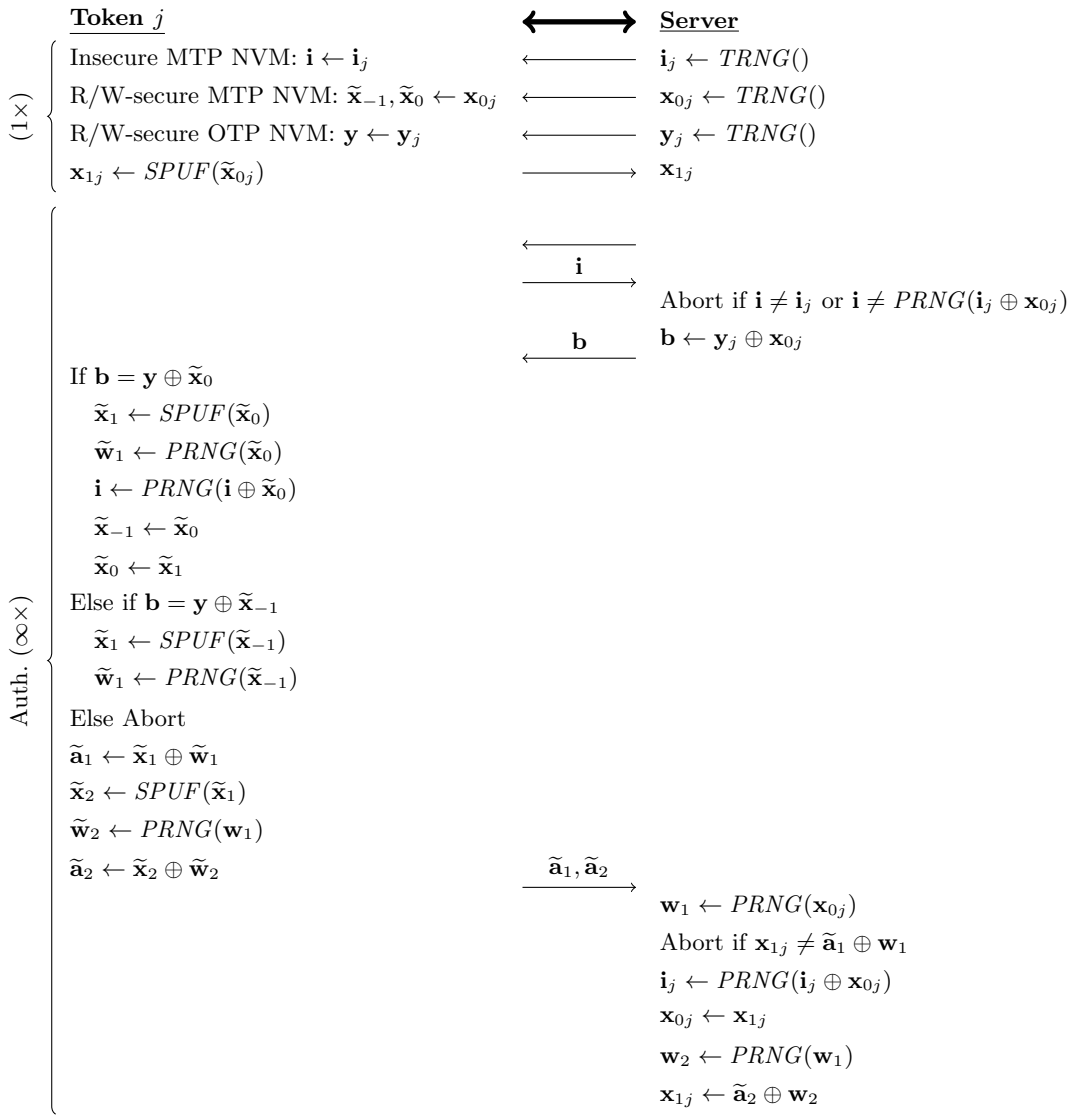


Fig. 8. Authentication protocol of Kulseng et al. We assume that $\mathbf{x_0}$ is used to update $\mathbf{i}$, given the contradictive specification.

Incomplete Specification (#1) 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 [17] and opt for $\mathbf{x_0}$. Also, the authors do not acknowledge the need to expand the response space of the strong PUF.

Secure NVM Undermines PUF Benefit (#2) The need for secure NVM undermines the main benefit of PUF technology. Either read or write access would enable an attacker to break the system.

Non-Functional due to PUF Noisiness (#3) 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 (#8) The desynchronization recovery logic allows for a simple replay attack. One can impersonate the server by resending the last $\mathbf{b}$.

Denial-of-Service (#8) As observed in [17], 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 (#8) The PRNG is instantiated with a LFSR. In [17], 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.

5.3 Sadeghi et al. (October 2010)

The protocol of Sadeghi et al. [39] is represented by Figures 1(e) and 9. Essentially, one uses a strong PUF to generate a cryptographic key. The authors claim token privacy, in particular for an attacker which obtains the NVM contents.

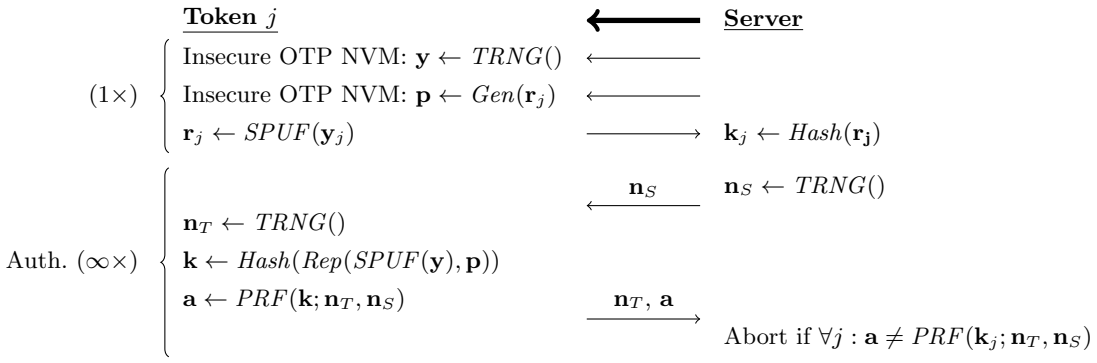


Fig. 9. Authentication protocol of Sadeghi et al. Unlike the proposing manuscript, we represent the fuzzy extractor as an explicit part of the protocol. We argue that implicit usage does not properly reflect the security and efficiency concerns.

Incomplete Specificiation (#1) The authors do not suggest the use of a particular strong PUF. The need to expand its response space is not acknowledged either.

Suboptimal (#6) Two inefficiencies have not been addressed. First, there is no need for the PUF to have an enormous input-output space. Even a weak PUF could be used and the NVM storing $\mathbf{y}$ could be omitted as well. Second, joint optimization of the fuzzy extractor and the authentication logic has not been explored, unlike Reference II-A which reuses cryptographic functions.

Privacy without Identification (#9) The proposal does not scale in the number of tags. One cannot introduce a public identifier as this would oppose the privacy claim.

5.4 Lee et al. I (March 2012)

The first protocol of Lee et al. [25] is represented by Figures 6(j) and 10. The authors instantiate *Enc* with the affiliated stream cipher NLM-128, with $\widetilde{\mathbf{r}}$ and $\mathbf{y}$ as key and initialization vector respectively. We note that *Enc* corresponds to key stream generation only rather than encryption, as there is no message involved. NVM physical security is not explicitly addressed, although we note that one can obtain $\mathbf{i}$ and $\mathbf{x} = \mathbf{a} \oplus \mathbf{b}$ via a direct query and eavesdropping respectively.

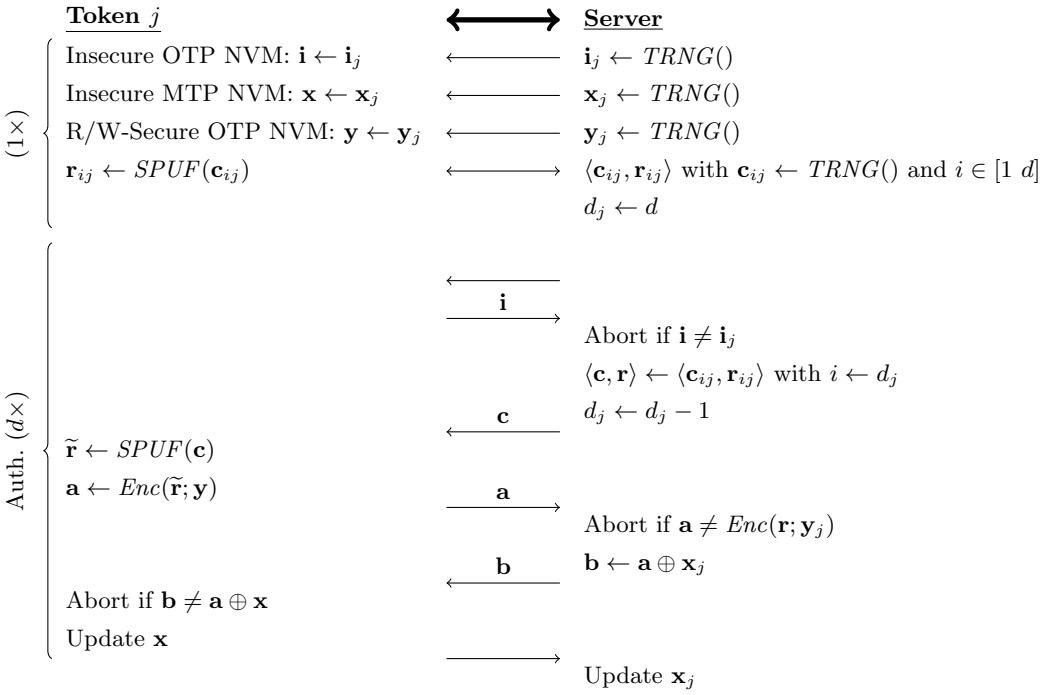


Fig. 10. Authentication protocol of Lee et al.

Incomplete Specification (#1) The update mechanism for $\mathbf{x}$ is not further specified, but should be chosen carefully. We argue that secret variable $\mathbf{y}$ should be involved in the update: $\mathbf{x} \leftarrow f(\mathbf{x}, \mathbf{y})$, rather than $\mathbf{x} \leftarrow f(\mathbf{x})$. If not, one could mount a simple server impersonation attack. Consider eavesdropping on a genuine protocol run (1). An attacker could get authenticated with an arbitrary $\mathbf{c}^{(2)}$ and $\mathbf{b}^{(2)} \leftarrow \mathbf{a}^{(2)} \oplus f(\mathbf{a}^{(1)} \oplus \mathbf{b}^{(1)})$. Also, the authors do not suggest the use of particular strong PUF. They do not acknowledge the need to expand its response space either.

Secure NVM Undermines PUF Benefit (#2) The use of secure NVM undermines the main benefit of PUF technology: improved physical security.

Non-Functional due to PUF Noisiness (#3) The authors mention the existence of fuzzy extractors as an error-correction mechanism. However, *Gen*/*Rep* procedures are not part of the proposed protocol and the need for helper data is not acknowledged either. We assume the complete absence of error-correction rather than an implicit usage, as security and efficiency concerns would not be reflected properly in the latter case. In practice, there will then be a mismatch $\widetilde{\mathbf{r}} = \mathbf{r} \oplus \mathbf{e}$, with $HW(\mathbf{e})$ relatively small, leading to token rejection. We make abstraction of the noisiness issue in the remainder of our analysis.

Denial-of-Service (#8) There is a simple denial-of-service attack. Either blocking or modifying the last message is sufficient to desynchronize token and server.

Cryptanalysis of NLM-128 (#8) A cryptanalytic attack on the stream cipher NLM-128 has recently been reported [32], so one might have to consider other ciphers.

5.5 Jin et al. (April 2012)

The proposal of Jin et al. [15] seems to be inspired by [23]. The protocol is represented by Figures 6(k) and 11. 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.

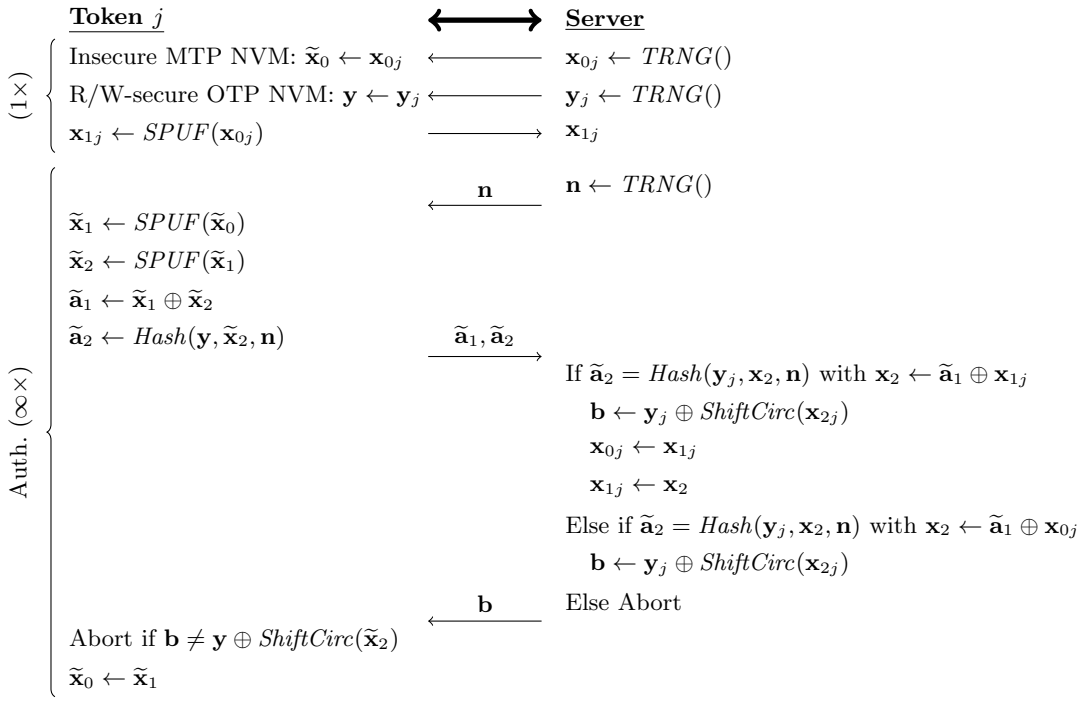


Fig. 11. Authentication protocol of Jin et al.

Incomplete Specification (#1) The authors instantiate the protocol with a custom variant of the feed-forward arbiter PUF. However, the description is too vague to allow for the interpretation of a workable PUF. Also, the need to expand its response space is not acknowledged.

Secure NVM Undermines PUF Benefit (#2) The use of secure NVM undermines the main benefit of PUF technology: improved physical security. 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 (#3) 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 (#8) 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 (#8) 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 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 $\widetilde{\mathbf{x}}_0$ (#8,#2) 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 (#9) The proposal does not scale in the number of tags. One cannot introduce a public identifier as this would oppose the privacy claim.

5.6 Xu et al. (September 2012)

The protocol of Xu et al. [46] is represented by Figures 6(m) and 12. There is a system key $\mathbf{y}$ which is shared by the server and all tokens. For improved privacy, token identifier $\mathbf{i}$ is updated after every authentication: 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.

Incomplete Specification (#1) There is only a vague informal description of the desynchronization recovery logic. The authors state that the server retains old identifiers $\mathbf{i}_{-1j}$, without providing further details. We filled in the blanks to the best of our insights and exclude this part from cryptanalysis. Also, the authors do not suggest the use of particular strong PUF. They do not acknowledge the need to expand its response space either.

Secure NVM (#2) The need for secure NVM undermines the main benefit of PUF technology: 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 as $\mathbf{i}_0 = \mathbf{a} \oplus \mathbf{y}$.

Non-Functional due to PUF Noisiness (#3) 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 (#4) 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 $\langle \mathbf{c}, \mathbf{r}_1 \rangle$.

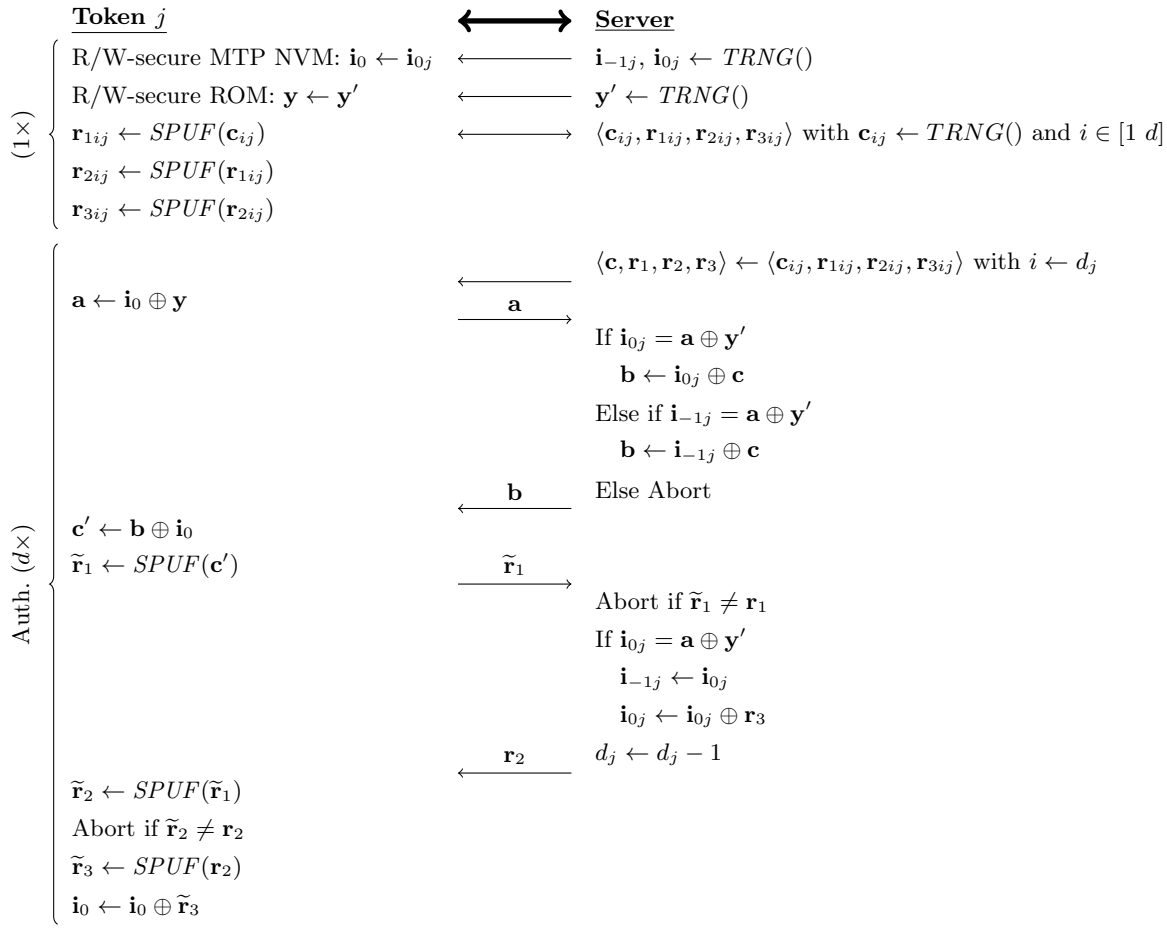


Fig. 12. Authentication protocol of Xu et al. The desynchronization recovery logic is reconstructed to the best of our insights, given its incomplete description.

Server Impersonation (#8) 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 $\mathbf{r}_2^{(2)} \leftarrow \mathbf{r}_2^{(1)}$.

5.7 He et al. (September 2012)

The protocol of He et al. [11] is represented by Figures 6(n) and 13. The PRNG is instantiated with a LFSR. For improved privacy, token identifier $\mathbf{i}$ is updated after every authentication: 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, the authors suggest the use of recovery logic to prevent denial-of-service from occurring. Exposure of the token state ($\mathbf{i}$, $\widetilde{\mathbf{x}}_0$ and $\mathbf{y}$) via physical leakage is claimed not to be a threat. The authors suggest the use of an arbiter PUF. The PUF error rate is assumed to be negligible.

Incomplete Specification (#1) There is only an incomplete and informal description of the desynchronization recovery logic. The authors suggest that tokens should retain their previous state, without providing further details. This by itself can never work, as the server might receive either an old or new identifier $\mathbf{i}$. Therefore, we are not able to fill in the blanks properly and exclude this part from cryptanalysis. However it should be noted that desynchronization recovery logic, when not carefully designed, often leads to one or more protocol flaws. Furthermore, the initialization of the state ($\mathbf{i}$, $\widetilde{\mathbf{x}}_0$ and $\mathbf{y}$) is not explicitly covered. Also, the authors do not acknowledge the need to expand the PUF response space.

PUF Noisiness Underestimated (#3) It is unrealistic to assume that the PUF error rate is negligible, making the protocol non-functional in practice. There will be a mismatch $\mathbf{x}'_1 = \widetilde{\mathbf{x}}_1 \oplus \mathbf{e}$, with $HW(\mathbf{e})$ relatively small, leading to server rejection. The use of $\widetilde{\mathbf{x}}_2$ and $\widetilde{\mathbf{x}}_3$ 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.

Denial-of-Service (#8) There are two simple denial-of-service attacks, both interfering with a genuine protocol run in order to desynchronize token and server. A first attack comprehends the modification of $\widetilde{\mathbf{a}}_2$ to an arbitrarily chosen value. A second attack exploits the LFSR linearity, by injecting $\mathbf{b}_1 \leftarrow \mathbf{b}_1 \oplus \mathbf{e}$ and $\mathbf{b}_2 \leftarrow \mathbf{b}_2 \oplus \mathit{LFSR}(\mathbf{e})$, with an arbitrary $\mathbf{e} \neq \mathbf{0}$.

Token/Server Impersonation, Denial-of-Service and Privacy Breach (#8) The use of a LFSR leads to full system disclosure. Assume that an attacker eavesdropped on a genuine protocol run, hereby obtaining $\mathbf{i}^{(1)}$, $\mathbf{b}_1^{(1)}$, $\mathbf{b}_2^{(1)}$, $\mathbf{a}_1^{(1)}$ and $\mathbf{a}_2^{(1)}$. Subsequently, one obtains $\mathbf{i}^{(2)}$, either by eavesdropping or a direct query. This allows for the retrieval of $\mathbf{n}_5^{(1)} = \mathbf{i}^{(1)} \oplus \mathbf{i}^{(2)}$. Given an n-bit output sequence of an n-bit LFSR, one can easily retrieve the initial state. This leads to $\mathbf{n}_4^{(1)}$, $\mathbf{n}_3^{(1)}$, $\mathbf{n}_2^{(1)}$ and $\mathbf{n}_1^{(1)}$ and hence also $\mathbf{x}_3^{(1)} = \mathbf{n}_4^{(1)} \oplus \mathbf{a}_1^{(1)}$, $\mathbf{x}_2^{(1)} = \mathbf{n}_3^{(1)} \oplus \mathbf{a}_2^{(1)}$, $\mathbf{x}_1^{(1)} = \mathbf{n}_2^{(1)} \oplus \mathbf{b}_2^{(1)}$ and $\mathbf{y}^{(1)} = \mathbf{n}_1^{(1)} \oplus \mathbf{b}_1^{(1)}$, respectively. From this point onwards, all secret variables of protocol run (2) are trivial to derive. Token impersonation extends to an unlimited number of of authentications: the PUF operation can be replaced with an arbitrary function.

Token/Server Impersonation, Denial-of-Service and Privacy Breach via Leakage (#8) We argue that leakage leads to full system disclosure, independent of the PRNG instantiation. Assume that an attacker retrieved $\widetilde{\mathbf{x}}_0$ and $\mathbf{y}$ by physical means. The state vector $\mathbf{i}$ can be obtained by a simple query. Subsequently, one eavesdrops on a genuine protocol run, hereby obtaining $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{a}_1$ and $\mathbf{a}_2$. This allows for the retrieval of $\mathbf{n}_1 = \mathbf{b}_1 \oplus \mathbf{y}$ and hence also $\mathbf{n}_2$ to $\mathbf{n}_6$ by repeated PRNG evaluation. Further implications are identical to the latter part of Section 5.7.

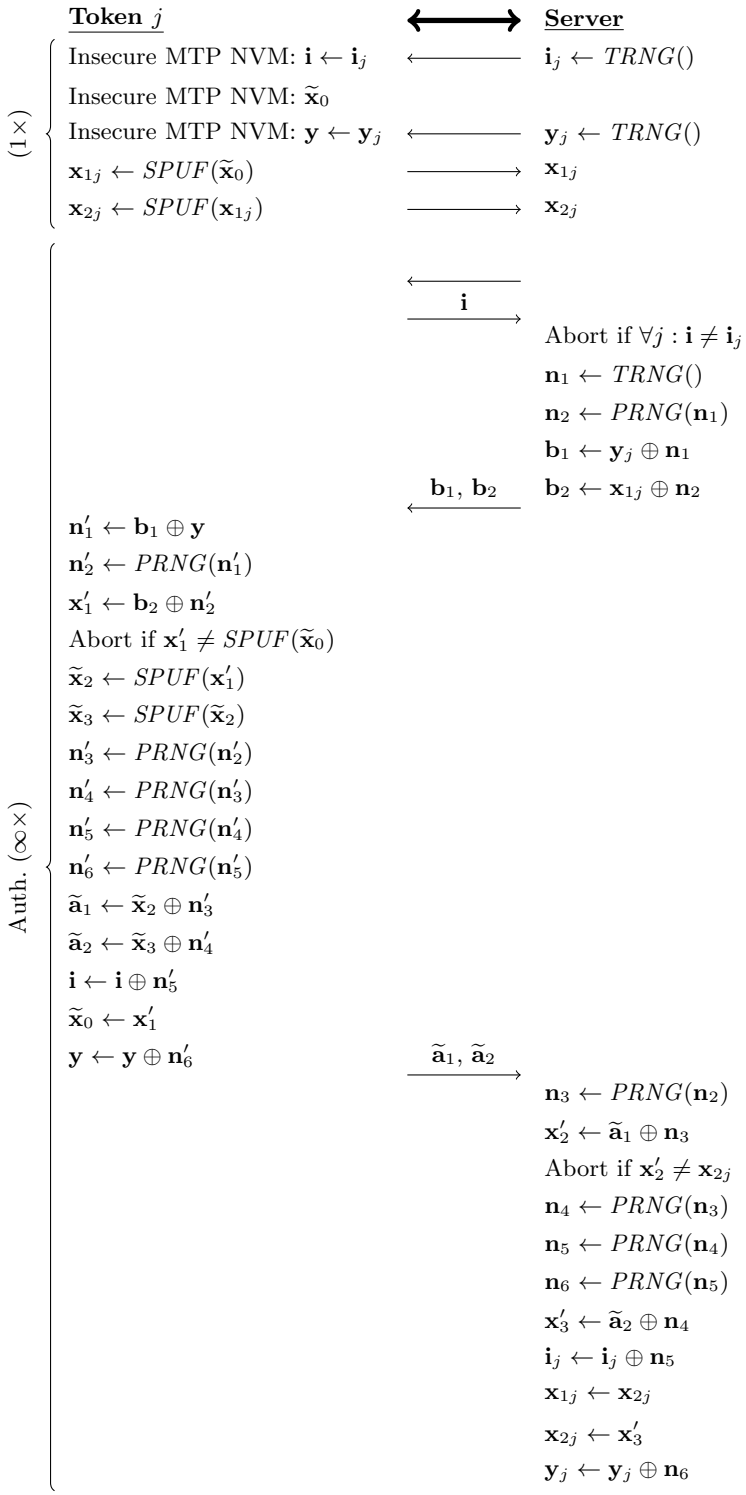


Fig. 13. Authentication protocol of He et al. Desynchronization recovery logic is not included due to its incomplete specification.

5.8 Jung et al. (January 2013)

The protocol of Jung et al. [16] is represented by Figures 6(o) and 14. The authors make use of timestamps at both sides. The authors make use of the HMAC standard [22], with the PUF response as key. There is a privacy claim: tracking is only possible in between protocol runs. Also, physical leakage of $\mathbf{c}_0$ is claimed not to be a threat. We note leakage of $\mathbf{c}_0$ and $\mathbf{i}_T$ to be equivalent, as one can query for $\mathbf{a}_1$.

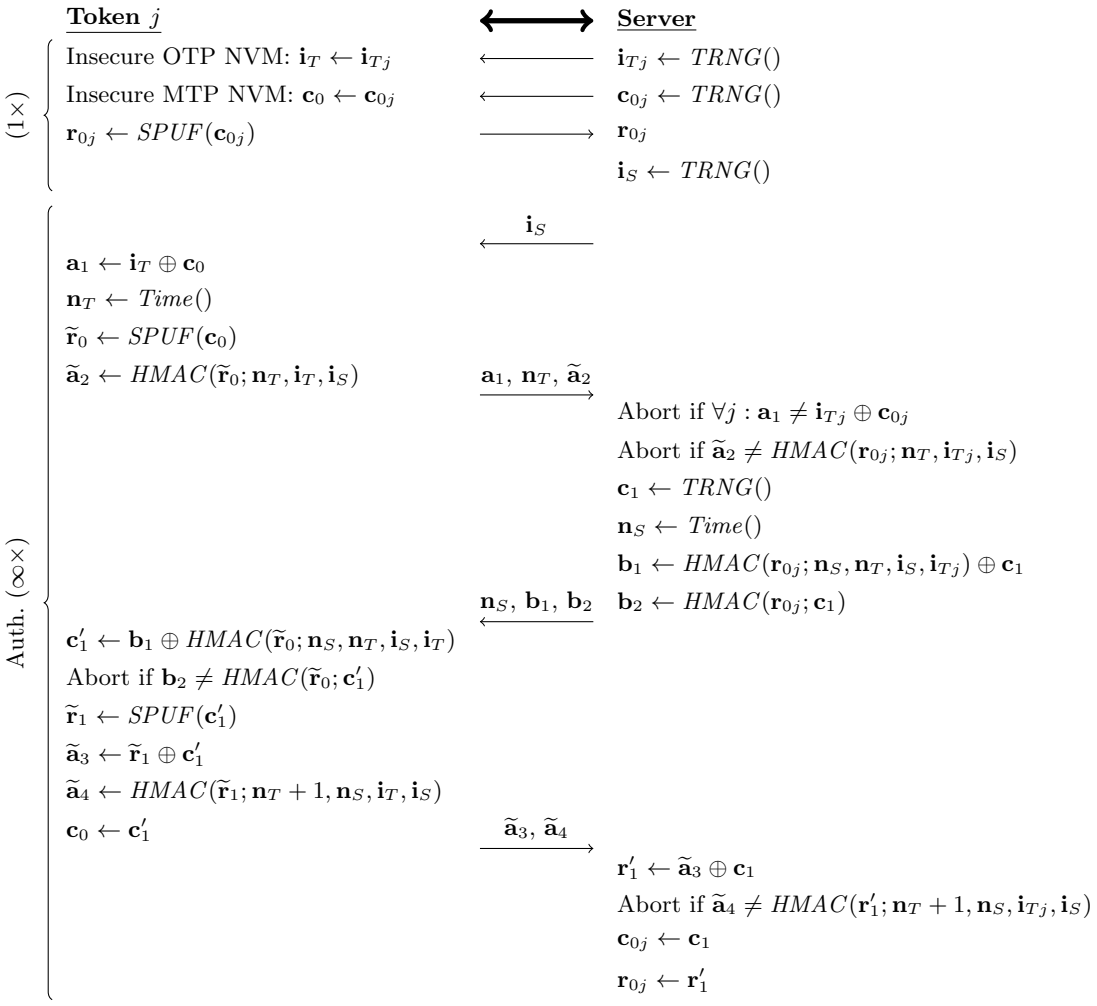


Fig. 14. Authentication protocol of Jung et al.

Incomplete Specification (#1) An implementation of $Time()$ has not been specified, which is crucial at the token side in particular. The timestamp could be a snapshot of either a synchronized clock or a local clock. One could also opt for a monotonic counter, incremented after every authentication, but this would require write-secure MTP NVM. However, given that in the protocol specification, neither party verifies the other party's timestamp (not even to verify that the current timestamp is greater than the previous one), one could equally well opt for a TRNG. Also, the authors do not suggest the use of a particular strong PUF. The need to expand its response space is not acknowledged either.

Non-Functional due to PUF Noisiness (#3) PUF responses are not acknowledged to be noisy, making the protocol non-functional. There will be a mismatch $\widetilde{\mathbf{r}}_0 = \mathbf{r}_{0j} \oplus \mathbf{e}$, with $HW(\mathbf{e})$ relatively small, leading to token rejection. We make abstraction of the noisiness issue in the remainder of our analysis.

Denial-of-Service (#8) There is a simple denial-of-service attack. Either blocking or modifying the last message is sufficient to desynchronize token and server.

Token/Server Impersonation, Denial-of-Service and Privacy Breach via Leakage (#8) We argue that leakage leads to full system disclosure. Assume an attacker to eavesdrop on a single protocol run, hereby obtaining $\mathbf{a}_3^{(1)}$. Subsequently, $\mathbf{i}_T$ and $\mathbf{c}_0^{(2)}$ are obtained via physical leakage. One can then retrieve $\mathbf{r}_0^{(2)} = \mathbf{a}_3^{(1)} \oplus \mathbf{c}_0^{(2)}$. An attacker can now replace the PUF with an arbitrary function and take control.

5.9 Lee et al. II (July 2013)

The second protocol of Lee et al. [26] is represented by Figures 6(p) and 15. Exposure of the token state ($\mathbf{i}$, $\mathbf{c}_0$ and $\mathbf{y}$) via physical leakage is claimed not to be a threat.

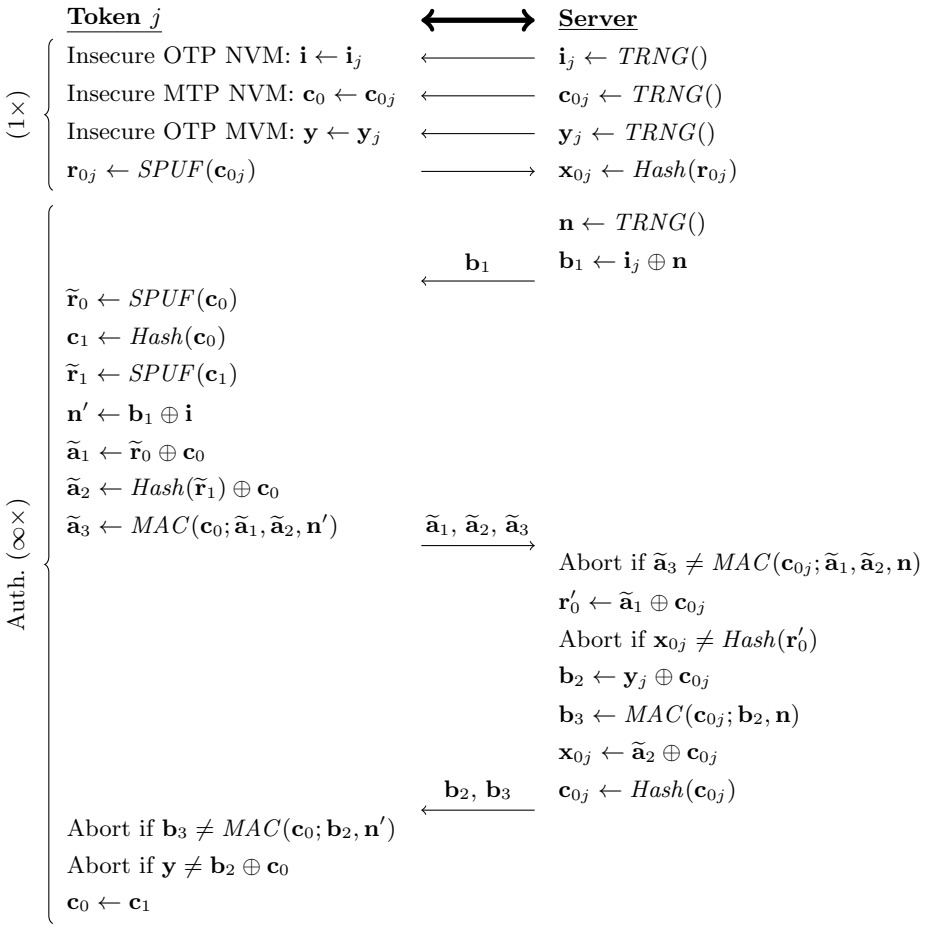


Fig. 15. Authentication protocol of Lee et al. We assume $\mathbf{c}_{0j}$ to be the key of MAC $\mathbf{b}_3$, as this has not been specified clearly.

Incomplete Specification (#1) The key for MAC $\mathbf{b}_3$ is not explicitly given, although there seems to be no objection against the use of $\mathbf{c}_{0j}$. Also, the authors do not suggest the use of particular strong PUF. They do not acknowledge the need to expand its response space either.

Non-Functional due to PUF Noisiness (#3) PUF responses are not acknowledged to be noisy, making the protocol non-functional. There will be a mismatch $\widetilde{\mathbf{r}}_0 = \mathbf{r}'_0 \oplus \mathbf{e}$, with $HW(\mathbf{e})$ relatively small, leading to token rejection. We make abstraction of the noisiness issue in the remainder of our analysis

No Guidance on Resources (#6) Neither the hash nor the MAC is instantiated. However, in order to save resources, it might be advisable to rely on a single cryptographic algorithm. E.g., a hash-based implementation of the MAC could be a possibility.

Denial-of-Service (#8) There is a simple denial-of-service attack. Either blocking or modifying the last message is sufficient to desynchronize token and server.

Token/Server Impersonation and Denial-of-Service via Leakage (#8) We argue that leakage leads to full system disclosure. Assume an attacker to observe a genuine protocol run (1). Via state leakage, one subsequently obtains $\mathbf{i}$, $\mathbf{c}_0^{(2)}$ and $\mathbf{y}$. First consider impersonation of the server, meanwhile causing desynchronization for the genuine token. One only has to select an arbitrary nonce $\mathbf{n}^{(2)}$ and compute $\mathbf{b}_1^{(2)}$, $\mathbf{b}_2^{(2)}$ and $\mathbf{b}_3^{(2)}$ accordingly. Now consider token impersonation, meanwhile causing desynchronization for the genuine token. An attacker has to interfere with a genuine protocol run (2). First, one retrieves $\mathbf{n}^{(2)} = \mathbf{b}_1^{(2)} \oplus \mathbf{i}$. The PUF is replaced with an arbitrary function which outputs $\widetilde{\mathbf{r}}_1$. One computes $\widetilde{\mathbf{a}}_2$ and $\widetilde{\mathbf{a}}_3$ accordingly. There is no modification to $\widetilde{\mathbf{a}}_1$.

5.10 Noise Bifurcation (May 2014)

The noise bifurcation proposal [47] can be understood as a minor variation on the slender PUF protocol, as also reflected by Figure 6(q). 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. 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 16 represents the protocol.

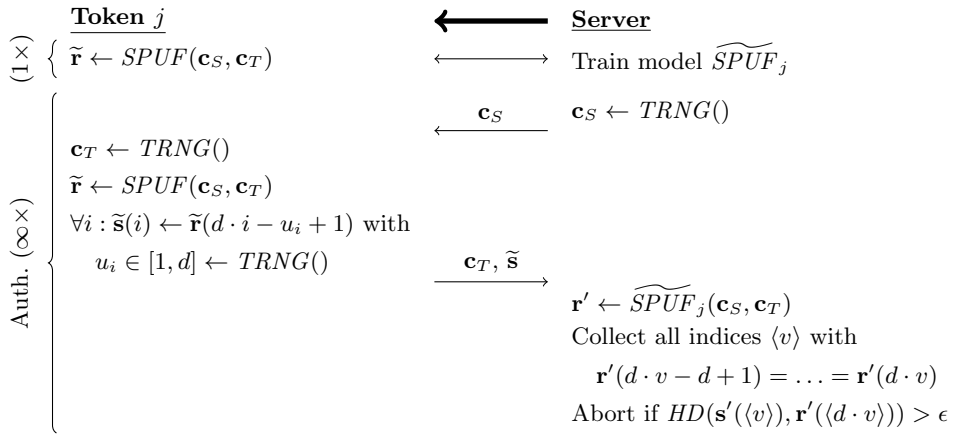


Fig. 16. Noise bifurcation protocol.

Incomplete Specification (#1) The protocol employs a PRNG to expand the PUF response space: $\widetilde{\mathbf{r}} \leftarrow SPUF(PRNG(\mathbf{c}_S, \mathbf{c}_T))$. 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.

Exacting PUF Requirements (#7) 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, the authors showed how to defeat the protocol using simulated arbiter XOR PUFs with up to four chains.

5.11 System of PUFs (October 2014)

The system of PUFs proposal [21] consists of three PUFs, as shown in Figure 6(r). 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 17 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.

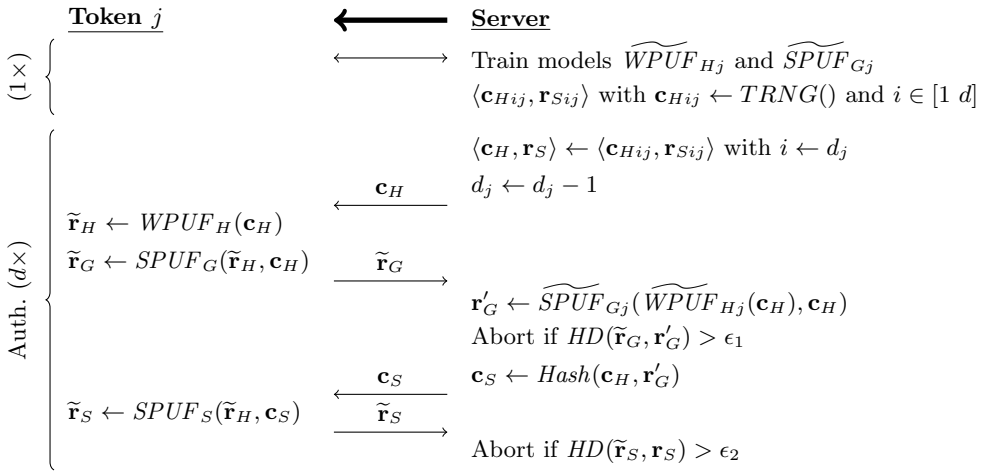


Fig. 17. System of PUFs.

PUF Noisiness Underestimated (#3) 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 [42], 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 (#4) Although the authors seem to suggest the opposite, there is no practical PUF which is robust against modeling. E.g., noisiness provides 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 (#6) 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 (#8) 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 (#8) The server depletion statements are unfair. It all depends on which party initiates the protocol, an aspect which has not been covered. 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.

6 Overview and Discussion

Table 1 provides an overview of Section 5. We adopt the perspective of an interested **system provider, aiming to select a protocol**. Proposals which do not offer any robustness against both noise (#3) and modeling (#4) are discarded first. Subsequently, we discard proposals which are vulnerable to conventional protocol attacks (#8). Despite the exploitation of their response expansion method (#5), 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 eight remaining proposals are marked bold for further consideration. We do not object that the discarded proposals might contain worthwhile concepts.

The eight 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. To save resources, the latter primitive might be reused as part of the key generation logic. All this might provide excellent security, but it is unfortunately not so very lightweight (#6). 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 (#7).

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 requests that require strong cryptographic primitives, e.g., message confidentiality and integrity, PUF-based key generation seems more appropriate.

7 Conclusion

Various protocols utilize a strong PUF to provide lightweight entity authentication. We described nineteen proposals using a unified notation, hereby creating a large-scale overview and initializing direct comparison as well. Our framework of protocol requirements considerably aided the analysis, revealing 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

		Reference II-A	**Reference II-B**	*Basic*	**Controlled**	*Bolotnyy et al.*	*Öztürk et al.*	*Hammouri et al.*	*Kulseng et al.*	**Sadeghi et al.**	*Reconfiguration*	**Reverse FE I**	**Reverse FE II**	*Converse*	*Lee et al. I*	*Jin et al.*	**Slender**	*Xu et al.*	*He et al.*	*Jung et al.*	*Lee et al. II*	**Noise bifurcation**	*System of PUFs*
resources (#6)	Weak PUF	✓	✓	×	×	×	×	×	×	×	×	×	✓	×	×	×	×	×	×	×	×	×	✓
	Strong PUF[1]	×	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	MTP NVM	×	×	×	×	✓	✓	×	✓	×	✓	×	×	×	✓	✓	×	✓	✓	✓	✓	×	×
	TRNG	×	✓	×	×	×	×	✓	×	✓	×	×	✓	✓	×	×	✓	×	×	∼	×	✓	×
	Gen	×	×	×	×	×	×	×	×	×	×	✓	✓	×	×	×	×	×	×	×	×	×	×
	Rep	✓	✓	×	×	×	×	×	×	✓	×	×	×	✓	×	×	×	×	×	×	×	×	×
	Crypto (Enc/Hash/MAC)	✓	✓	×	✓	×	×	×	×	✓	✓	✓	✓	✓	✓	✓	×	×	×	✓	✓	×	×
	PRNG	×	×	×	×	×	×	×	✓	×	×	×	×	×	×	×	×	×	✓	×	×	×	×
	$\oplus, =?$	×	✓	×	×	×	✓	✓	✓	×	×	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	×	×
	$1\times$ interface	✓	✓	×	✓	×	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×
claims	Server authenticity	×	✓	×	×	×	×	×	✓	×	×	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	×	×
	Token authenticity	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Token privacy	×	×	×	×	✓	×	×	✓	✓	∼	×	×	×	×	✓	×	✓	✓	✓	×	×	×
	Leakage resilience (#2)					×	×		×	✓	∼			✓	∼		×	✓	✓	✓			
	# Authentications	∞	∞	d	d	d	∞	∞	∞	∞	d	∞	∞	∞	d	∞	∞	d	∞	∞	∞	∞	d
evaluation	Noise Robust (#3)	✓	✓		✓	×	∼	×	×	✓		✓	✓		×	×	✓	×	×	×	×	✓	∼
	Modeling Robust (#4)	✓	✓	×	✓	×				✓	×	✓	✓				∼	∼				∼	×
	PUF Independency (#7)	✓	✓		∼					∼		×	✓				×					×	
	Server authenticity (#8)		✓						×			∼	✓	×	✓	×		×	×	×			
	Token authenticity (#8)	✓			✓				×	✓		∼	✓		✓		∼		×	×		✓	
	Token privacy (#8)								×	✓									×	×			
	DoS prevention (#8)	✓	✓		✓	×	×		×	✓	×	✓	✓		×	×	✓		×	×	×	✓	
	Leakage resilience (#8)									✓						×			×	×	×		
	Scalability[2] (#9)	✓	✓	✓	✓					×		✓	✓			×	✓					✓	

[1] Including logic to expand the PUF response (#5).

[2] Including a public token identifier stored in insecure OTP NVM for proposals which do not claim privacy.

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'.

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. The observations and lessons learned in this work can facilitate future protocol design.

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. R. Bassil, W. El-Beaino, A. I. Kayssi, and A. Chehab. A puf-based ultra-lightweight mutual-authentication RFID protocol. In *6th International Conference for Internet Technology and Secured Transactions, ICITST 2011, Abu Dhabi, UAE, December 11-14, 2011*, pages 495–499, 2011.
2. L. Bolotnyy and G. Robins. Physically unclonable function-based security and privacy in RFID systems. In *Fifth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2007), 19-23 March 2007, White Plains, New York, USA*, pages 211–220, 2007.
3. X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure remote authentication using biometric data. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 147–163, 2005.
4. J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede. Helper data algorithms for puf-based key generation: Overview and analysis. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, PP(99):1–1, 2014.
5. 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.
6. 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.
7. Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 523–540, 2004.
8. 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.
9. 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.
10. G. Hammouri, E. Öztürk, and B. Sunar. A tamper-proof and lightweight authentication scheme. *Pervasive and Mobile Computing*, 4(6):807–818, 2008.
11. Z. He and L. Zou. High-efficient rfid authentication protocol based on physical unclonable function. In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2012 8th International Conference on*, pages 1–4, Sept 2012.
12. 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.
13. 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.
14. 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.
15. 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.
16. S. W. Jung and S. Jung. Hrp: A hmac-based rfid mutual authentication protocol using puf. In *Information Networking (ICOIN), 2013 International Conference on*, pages 578–582, Jan 2013.

17. 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.
18. S. Kardaş, S. ÇElik, M. YildiZ, and A. Levi. Puf-enhanced offline rfid security and privacy. *J. Netw. Comput. Appl.*, 35(6):2059–2067, Nov. 2012.
19. S. Katzenbeisser, Ünal Koçabas, V. van der Leest, A. Sadeghi, G. J. Schrijen, H. Schröder, and C. Wachsmann. Recyclable pufs: Logically reconfigurable pufs. In *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pages 374–389, 2011.
20. Ünal. Kocabaş, A. Peter, S. Katzenbeisser, and A. Sadeghi. Converse puf-based authentication. In *Trust and Trustworthy Computing - 5th International Conference, TRUST 2012, Vienna, Austria, June 13-15, 2012. Proceedings*, pages 142–158, 2012.
21. 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.
22. H. Krawczyk, M. Bellare, and R. Canetti. Hmac: Keyed-hashing for message authentication. Technical report, 1997.
23. L. Kulseng, Z. Yu, Y. Wei, and Y. Guan. Lightweight mutual authentication and ownership transfer for rfid systems. In *INFOCOM*, pages 251–255, 2010.
24. 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, 2004.
25. Y. S. Lee, T. Y. Kim, and H. J. Lee. Mutual authentication protocol for enhanced rfid security and anti-counterfeiting. In *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*, pages 558–563, March 2012.
26. Y. S. Lee, H. J. Lee, and E. Alasaarela. Mutual authentication in wireless body sensor networks (wbsn) based on physical unclonable function (puf). In *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, pages 1314–1318, July 2013.
27. R. Maes. *Physically Unclonable Functions: Constructions, Properies and Applications*. PhD thesis, KU Leuven, 2012.
28. 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.
29. 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.
30. 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.
31. 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.
32. M. A. Orumiehchiha, J. Pieprzyk, and R. Steinfeld. Breaking nlm-mac generator. Cryptology ePrint Archive, Report 2013/202, 2013. http://eprint.iacr.org/.
33. E. Öztürk, G. Hammouri, and B. Sunar. Towards robust low cost authentication for pervasive devices. In *PerCom*, pages 170–178. IEEE Computer Society, 2008.
34. R. S. Pappu. *Physical One-Way Functions*. PhD thesis, MIT, 2001.
35. 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.
36. 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.
37. 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.
38. 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*, 2014.
39. A.-R. Sadeghi, I. Visconti, and C. Wachsmann. Enhancing rfid security and privacy by physically unclonable functions. In A.-R. Sadeghi and D. Naccache, editors, *Towards Hardware-Intrinsic Security*, Information Security and Cryptography, pages 281–305. Springer Berlin Heidelberg, 2010.
40. M. Safkhani, N. Bagheri, and M. Naderi. Security analysis of a PUF based RFID authentication protocol. *IACR Cryptology ePrint Archive*, 2011:704, 2011.

41. 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.
42. G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *DAC*, pages 9–14. IEEE, 2007.
43. 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.
44. P. Tuyls and L. Batina. Rfid-tags for anti-counterfeiting. In *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, pages 115–131, 2006.
45. 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.
46. 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.
47. 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.