A Survey on Lightweight Entity Authentication with Strong PUFs

Jeroen Delvaux[1,2], Roel Peeters[1], Dawu Gu[2] 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.

Keywords: physically unclonable function, entity authentication, lightweight

1 Introduction

In this work[1][2], 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 *electrically erasable programmable read-only memory* (EEPROM) and its successor Flash, battery-backed *static random-access memory* (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 *gate equivalents*, 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 [55] 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 *complementary metal-oxide-semiconductor* (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.

[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.

[2] Accepted for ACM Computing Surveys.

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 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 [45], controlled PUFs [15], Bolotnyy et al. [4], Öztürk et al. [44], Hammouri et al. [18], Kulseng et al. [33], Sadeghi et al. [53], logically reconfigurable PUFs [29], reverse fuzzy extractors [59], the converse protocol [30], Lee et al. I [35], Jin et al. [23], slender PUFs [40], Xu et al. [60], He et al. [19], Jung et al. [24], Lee et al. II [36], noise bifurcation [61] and system of PUFs [31].

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 [22] and the ring oscillator PUF[3] [56] are both very popular. Spatial correlations among cells are to be expected. Addressing the array provides a challenge-response mechanism. However, the total array size is of primary importance: mostly, all $2^m \cdot n$ 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 [11], 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 [34]. However, correlations among CRPs are severe, as these are not only spatial in nature but also induced by the functional behavior. Therefore, 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 [51].

Former subdivision comprehends a popular adaption of more restrictive notions in [17]. Originally, weak and strong PUFs were required to be tamper evident. Furthermore, correlations between CRPs were required to be negligible. For strong PUFs, this would implicate resistance against modeling attacks. Although highly relevant as an ideal case specification, we stick to the more recent practical notions.

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 [3, 20, 41, 52, 57] 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 [8], secure sketches [11, 12] are a particularly useful tool, as defined by a two-step procedure. First, public helper data is generated: $\mathbf{p} = \mathit{Gen}(\mathbf{r})$. Second, reproduction is performed: $\mathbf{r} = \mathit{Rep}(\widetilde{\mathbf{r}}, \mathbf{p})$. 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 in Figure 1. Both the code-offset and syndrome construction employ a binary $[n, k, t]$ block code $\mathcal{C}$, with t the error-correcting capability.

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

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* [59].

(a)

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 $\mathbf{w}$
	$\mathbf{r} \leftarrow \mathbf{p} \oplus \mathbf{w}$

(b)

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

Fig. 1. Secure sketch constructions: (a) code-offset, (b) syndrome.

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. It is advisable to extend all of the former with a helper data integrity check [5]. E.g., a token could check whether $Hash(\mathbf{r}, \mathbf{p})$ matches a reference value stored in NVM. This disables the exploitation of token failure rate probabilities, which has been identified as a threat for a variety of helper data methods [8]. However, this does not stop the enhancement of side channel analysis via helper data manipulation, as in [25] and [42].

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. The interpretation of individual items ranges from strictly required to highly desired, as nuanced hereafter.

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 [38].

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 [34,39,50,51]. 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 [17], as has been observed by others, e.g., [37]. With strong cryptographic primitives such as a hash function, one can fully mitigate this threat. Several proposals opt for more lightweight logic, such as XORing, a pseudorandom number generator (PRNG), a true random number generator (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 [50] 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 [39]. 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 [29, 30]. 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 [18]. A small PRNG, such as a linear feedback shift register (LFSR), is often employed [10, 18, 31, 40, 46, 59, 61]. Challenge $\mathbf{c}$ is then used as seed value: $\widetilde{\mathbf{r}} \leftarrow \mathit{SPUF}(\mathit{PRNG}(\mathbf{c}))$. In [44], a repeated permutation is employed for the same purpose. One might also be able to reuse cryptographic primitives, e.g., in [29], 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 is 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)

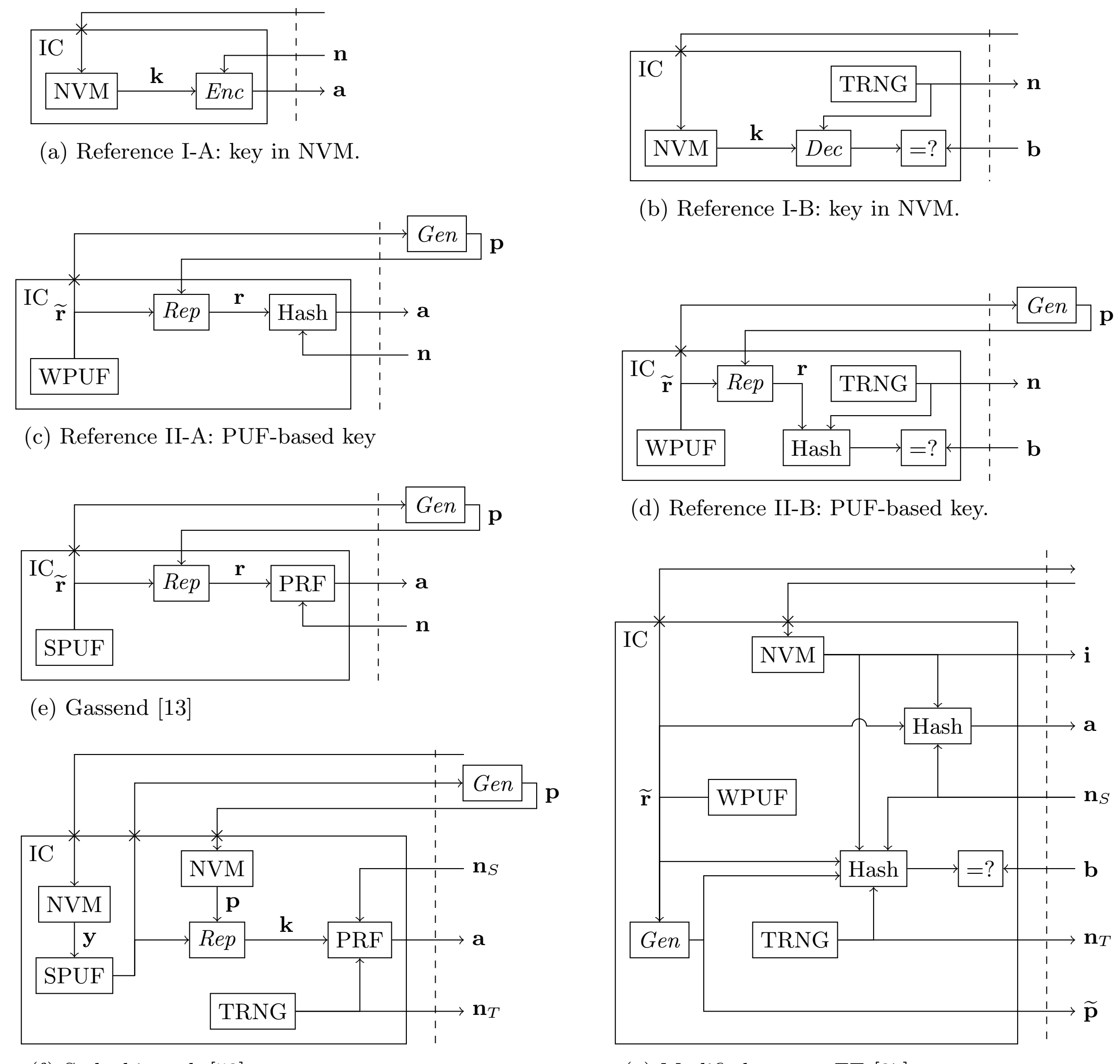
We distinguish between authentication and identification. The former comprehends the secure verification of an identity while the latter is limited to an unverified claim. Most protocols in this work provide token authentication and therefore also token identification. Unfortunately, identification prior 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 [59]. 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 [21]. 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



(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) Gassend [13]

(f) Sadeghi et al. [53].

(g) Modified reverse FE [37].

Fig. 2. 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 ×.

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 2, we now refine foregoing sketch on concatenated protocols.

4.1 Reference Protocols

Reference I-A in Figure 2(a) relies on physically secure NVM, an assumption which might not always hold in practice. Figure 3 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 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 2(b) and 4, provides server authenticity. Both mechanisms could be combined in order to provide mutual authentication.

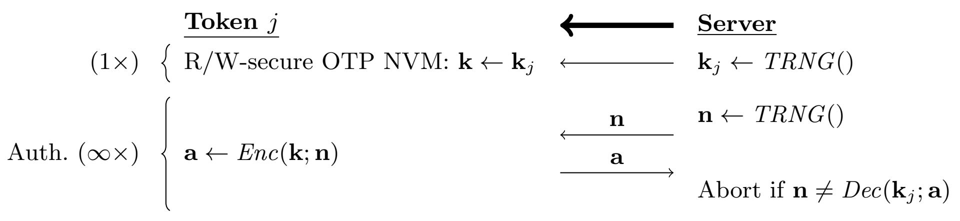

Token j
Server
$(1\times)$
R/W-secure OTP NVM: $\mathbf{k} \leftarrow \mathbf{k}_j$
$\mathbf{k}_j \leftarrow \mathit{TRNG}()$
Auth. $(\infty\times)$
$\mathbf{n} \leftarrow \mathit{TRNG}()$
$\mathbf{n}$
$\mathbf{a} \leftarrow \mathit{Enc}(\mathbf{k};\mathbf{n})$
$\mathbf{a}$
Abort if $\mathbf{n} \neq \mathit{Dec}(\mathbf{k}_j;\mathbf{a})$

Fig. 3. Authentication with Reference I-A. The thick arrow points from verifier to prover.

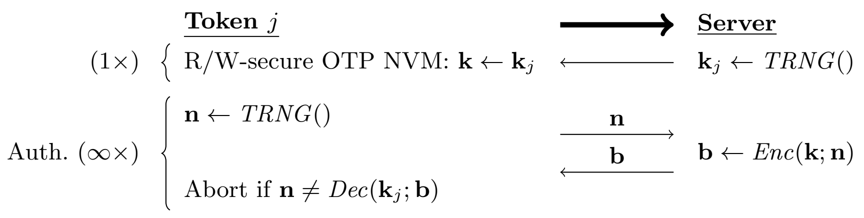

Token j
Server
$(1\times)$
R/W-secure OTP NVM: $\mathbf{k} \leftarrow \mathbf{k}_j$
$\mathbf{k}_j \leftarrow \mathit{TRNG}()$
Auth. $(\infty\times)$
$\mathbf{n} \leftarrow \mathit{TRNG}()$
$\mathbf{n}$
$\mathbf{b}$
$\mathbf{b} \leftarrow \mathit{Enc}(\mathbf{k};\mathbf{n})$
Abort if $\mathbf{n} \neq \mathit{Dec}(\mathbf{k}_j;\mathbf{b})$

Fig. 4. Authentication with Reference I-B.

Reference II-A, represented by Figures 2(c) and 5, 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. Strong PUFs typically require less area to provide a given number of response bits. The other side of the coin is that these bits are more correlated, requiring more bits to provide identical entropy and hence increasing the post-processing burden 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. For simplicity, we did omit a helper data integrity check, although we highly recommend to include one. One could generate a key as $\mathbf{k} \leftarrow \mathit{Hash}(\mathbf{r})$. We perform an optimization by merging this step with the authentication. Reference II-B, represented by Figure 2(d) and 6, provides server authenticity. Again, both mechanisms could be combined in order to provide mutual authentication.

4.2 Existing Literature

Several proposals in literature aim to provide entity authentication via PUF-based key generation, as listed next. The pioneering work of Gassend [13], widely praised for introducing the first silicon PUF, uses the

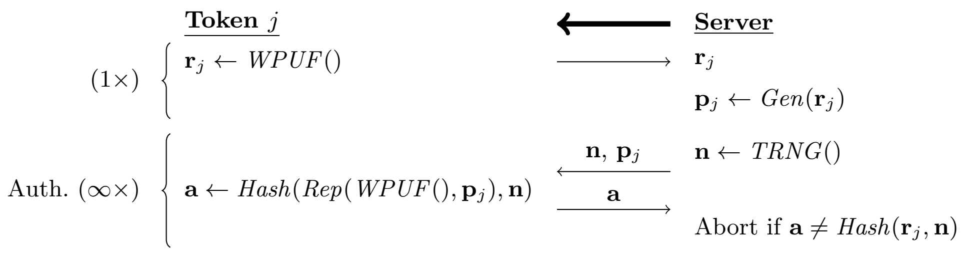

Token j
Server
$(1\times)$
$\mathbf{r}_j \leftarrow WPUF()$
$\mathbf{r}_j$
$\mathbf{p}_j \leftarrow Gen(\mathbf{r}_j)$
Auth. $(\infty\times)$
$\mathbf{n}, \mathbf{p}_j$
$\mathbf{n} \leftarrow TRNG()$
$\mathbf{a} \leftarrow Hash(Rep(WPUF(), \mathbf{p}_j), \mathbf{n})$
$\mathbf{a}$
Abort if $\mathbf{a} \neq Hash(\mathbf{r}_j, \mathbf{n})$

Fig. 5. Authentication with Reference II-A.

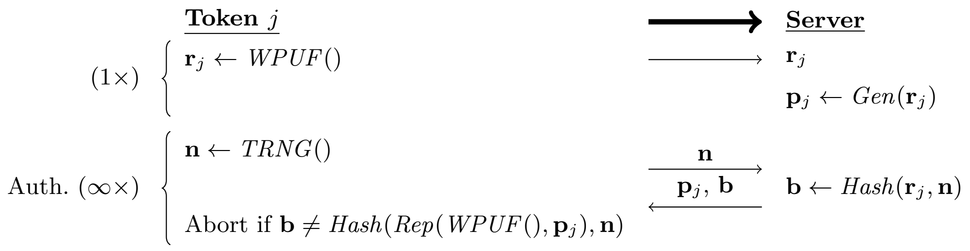

Token j
Server
$(1\times)$
$\mathbf{r}_j \leftarrow WPUF()$
$\mathbf{r}_j$
$\mathbf{p}_j \leftarrow Gen(\mathbf{r}_j)$
Auth. $(\infty\times)$
$\mathbf{n} \leftarrow TRNG()$
$\mathbf{n}$
$\mathbf{p}_j, \mathbf{b}$
$\mathbf{b} \leftarrow Hash(\mathbf{r}_j, \mathbf{n})$
Abort if $\mathbf{b} \neq Hash(Rep(WPUF(), \mathbf{p}_j), \mathbf{n})$

Fig. 6. Authentication with Reference II-B.

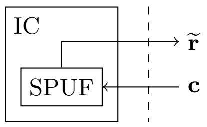
term *physically obfuscated key*. We represent the corresponding protocol by its block diagram in Figure 2(e) only, due to its incomplete specification and given the striking similarities with Reference II-A. Their strong PUF is evaluated with hardwired challenges, which would be equivalent with the later notion of weak PUFs. A pseudorandom function (PRF) is suggested as cryptographic primitive. Tuyls et al. [58] 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. [53] 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 [37] comprehends a special case of secret key generation, as the key is adapted to the noise. We analyze this protocol later-on, together with the original strong PUF version. There is the protocol of Bassil et al. [1], which has been badly broken in [54]. The proposal of Kardaş et al. [27] 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

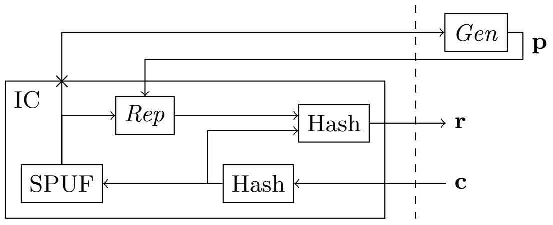
We describe and analyze all strong PUF protocols in chronological order, guided by Figure 7. As some protocols contain similar design concepts, we also considered categorizing them. However, it does not fly all the way, so we opt for the history of development as main theme instead. Sections 5.1 to 5.19 can be read in arbitrary order, although we recommend to get acquainted with the basic protocol first.

5.1 Basic Authentication (2001)

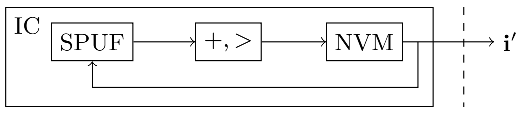
The most simple authentication method [6, 10, 45, 47] employs an unprotected strong PUF only, as shown in Figure 7(a). Figure 8 represents the corresponding protocol. For each token j, the server collects a database of d arbitrary CRPs during enrollment. A genuine token should be able to reproduce the response for each challenge in the server database. Only an approximate match is required, taking PUF noisiness into account: Hamming distance threshold ϵ implements this. To avoid token impersonation via replay, CRPs are discarded after use, limiting the number of authentications to d. As a direct consequence, the protocol should be strictly server-initiated. Otherwise an attacker might perform denial-of-service via exhaustion of the server database. By selecting, e.g., $m = 128$, an attacker cannot gather and tabulate all CRPs and clone a token as such. By selecting, e.g., $n = 128$, random guessing of $\mathbf{r}$ is extremely unlikely to succeed.



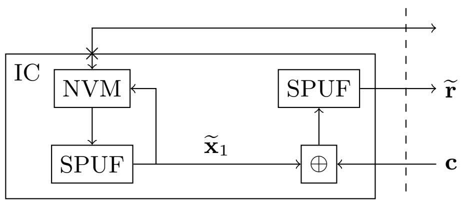
(a) Basic [45].



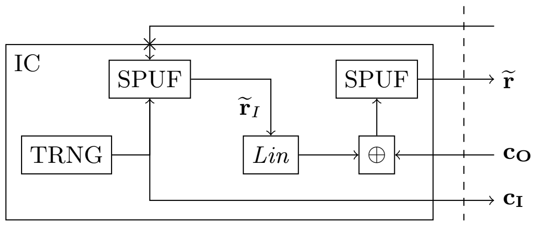
(b) Controlled PUF [15].



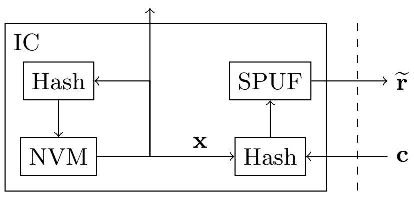
(c) Bolotnyy et al. [4].



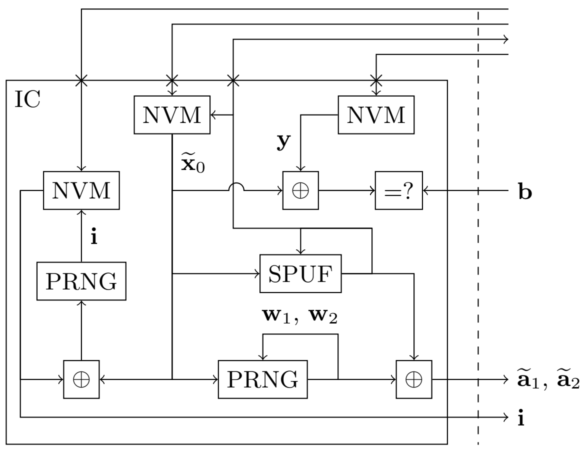
(d) Öztürk et al. [44].



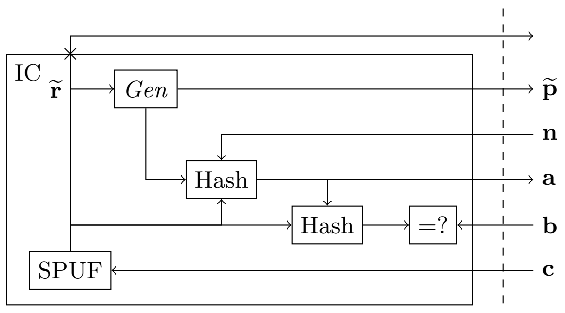
(e) Hammouri et al. [18].



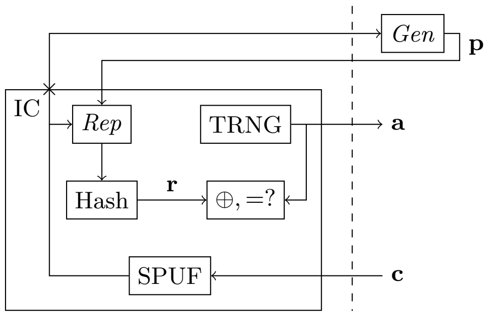
(g) Logically reconfigurable PUF [29].


IC
NVM
NVM
$\mathbf{y}$
$\widetilde{\mathbf{x}}_0$
NVM
⊕
=?
$\mathbf{b}$
$\mathbf{i}$
PRNG
SPUF
$\mathbf{w}_1$, $\mathbf{w}_2$
⊕
PRNG
⊕
$\widetilde{\mathbf{a}}_1$, $\widetilde{\mathbf{a}}_2$
$\mathbf{i}$

(f) Kulseng et al. [33].

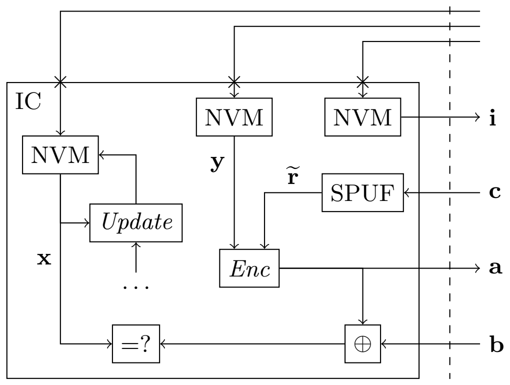


(h) Reverse FE [59].

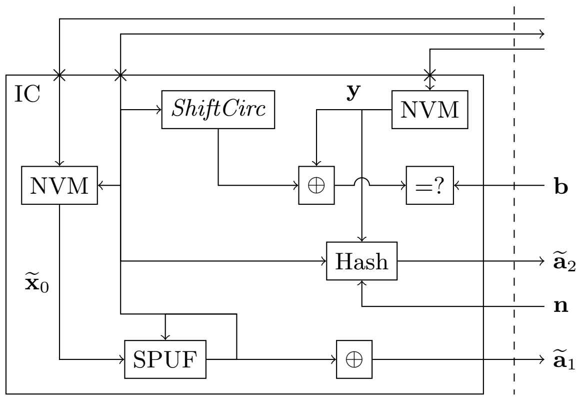


(i) Converse [30].

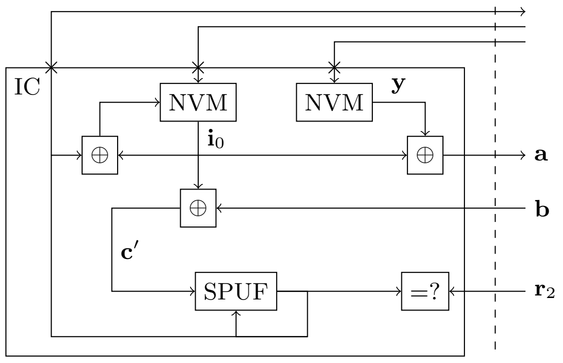
Fig. 7. Token representation for all strong PUF protocols. The following IC logic is not drawn: expansion of the strong PUF responses, intermediary registers (volatile) and control.



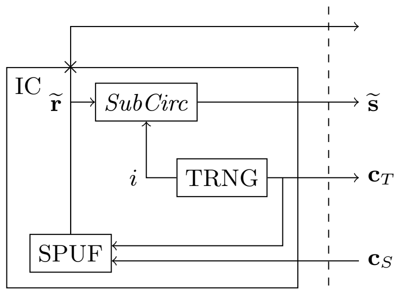
(j) Lee et al. I [35].



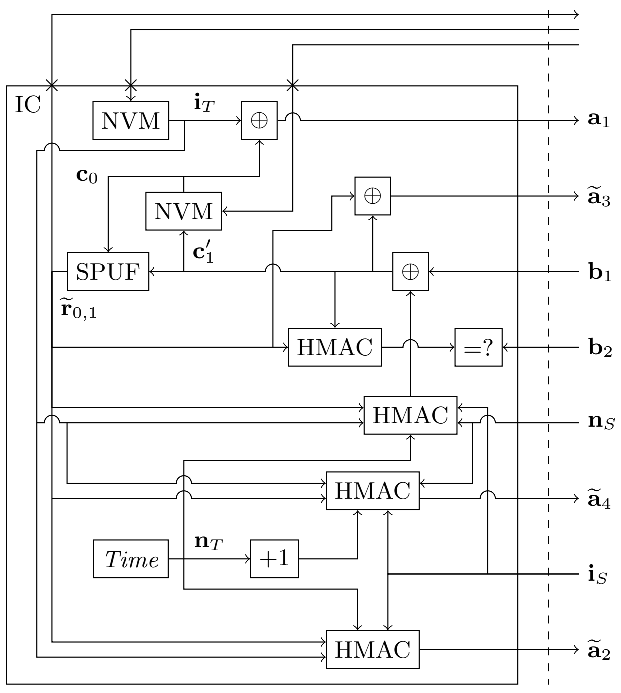
(k) Jin et al. [23].



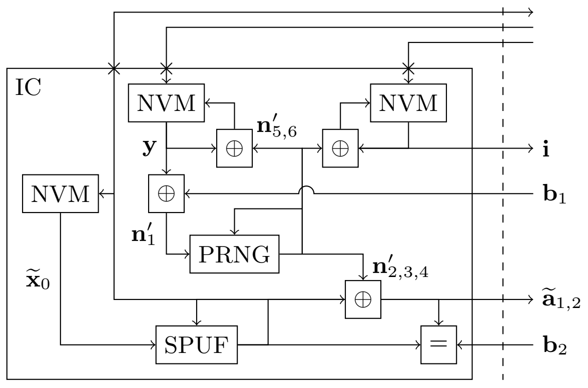
(m) Xu et al. [60].



(l) Slender PUF [40].

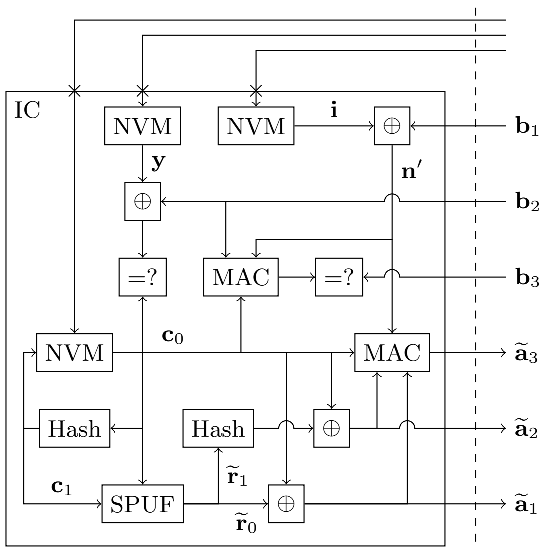


(o) Jung et al. [24].

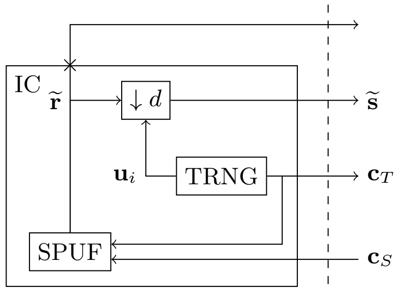


(n) He et al. [19].

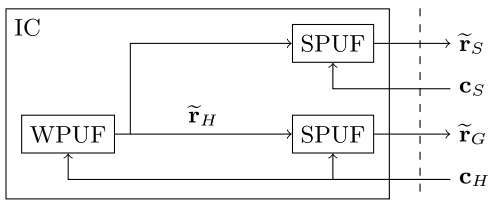
Fig. 7. Token representation for all strong PUF protocols (continued).



(p) Lee et al. II [36].



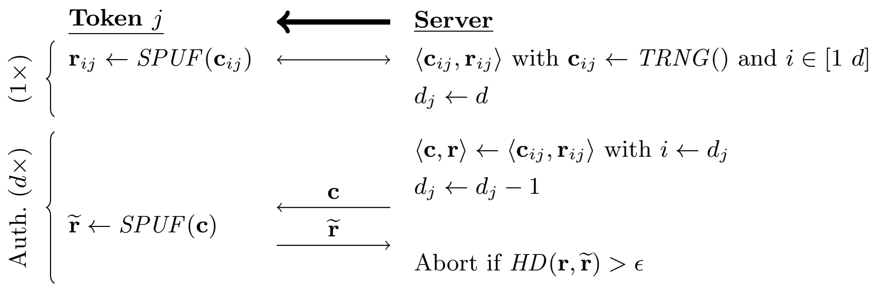
(q) Noise bifurcation [61].



IC
SPUF
$\widetilde{\mathbf{r}}_S$
$\mathbf{c}_S$
$\widetilde{\mathbf{r}}_H$
WPUF
SPUF
$\widetilde{\mathbf{r}}_G$
$\mathbf{c}_H$

(r) System of PUFs [31].

Fig. 7. Token representation for all strong PUF protocols (continued).



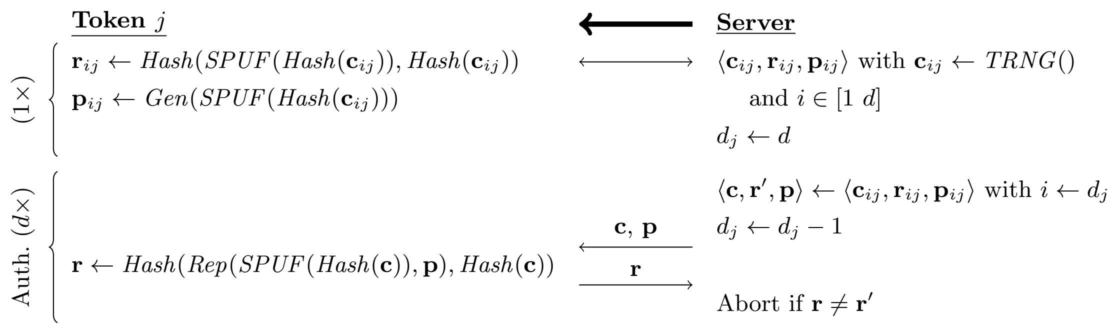
Token j
Server
$(1\times)$
$\mathbf{r}_{ij} \leftarrow SPUF(\mathbf{c}_{ij})$
$\langle \mathbf{c}_{ij}, \mathbf{r}_{ij} \rangle$ with $\mathbf{c}_{ij} \leftarrow TRNG()$ and $i \in [1\ d]$
$d_j \leftarrow d$
Auth. $(d\times)$
$\langle \mathbf{c}, \mathbf{r} \rangle \leftarrow \langle \mathbf{c}_{ij}, \mathbf{r}_{ij} \rangle$ with $i \leftarrow d_j$
$d_j \leftarrow d_j - 1$
$\mathbf{c}$
$\widetilde{\mathbf{r}} \leftarrow SPUF(\mathbf{c})$
$\widetilde{\mathbf{r}}$
Abort if $HD(\mathbf{r}, \widetilde{\mathbf{r}}) > \epsilon$

Fig. 8. Basic authentication protocol.

Modeling Attacks (#4) The proposal does not prevent PUF modeling attacks. A predictive model would enable token impersonation. Therefore, the practical value of the protocol is rather limited: there is no secure instantiation due to the lack of an appropriate strong PUF. We stress that the former applies to silicon PUFs. Optical PUFs [45], which initiated PUF history, are generally believed to be much more robust against modeling. Unfortunately, these are not so very compatible with low-cost integrated applications.

5.2 Controlled PUFs (November 2002)

Controlled PUFs [13–16] provide reinforcement against modeling via a cryptographic hash function (one-way). Two instances, preceding and succeeding the PUF respectively, are shown in Figure 7(b). Figure 9 represents the corresponding protocol. The preceding instance eliminates the chosen-challenge advantage of an attacker. The succeeding instance hides exploitable correlations due to the PUF structure. The latter hash seems to provide full protection by itself, but requires preceding error-correction: its avalanche effect would trigger on a single bit flip. As an optional reinforcement, the authors suggest to use the PUF in feedback mode. The error-corrected PUF output is then fed back as a PUF input, and this for multiple rounds. CRPs stored by the server are accompanied by helper data. The authors introduced the first silicon PUF, a predecessor of the arbiter PUF, the latter which makes history as the most popular strong PUF architecture.



Token j
Server
$(1\times)$
$\mathbf{r}_{ij} \leftarrow Hash(SPUF(Hash(\mathbf{c}_{ij})), Hash(\mathbf{c}_{ij}))$
$\mathbf{p}_{ij} \leftarrow Gen(SPUF(Hash(\mathbf{c}_{ij})))$
$\langle \mathbf{c}_{ij}, \mathbf{r}_{ij}, \mathbf{p}_{ij} \rangle$ with $\mathbf{c}_{ij} \leftarrow TRNG()$ and $i \in [1\ d]$
$d_j \leftarrow d$
Auth. $(d\times)$
$\langle \mathbf{c}, \mathbf{r}', \mathbf{p} \rangle \leftarrow \langle \mathbf{c}_{ij}, \mathbf{r}_{ij}, \mathbf{p}_{ij} \rangle$ with $i \leftarrow d_j$
$d_j \leftarrow d_j - 1$
$\mathbf{c}, \mathbf{p}$
$\mathbf{r} \leftarrow Hash(Rep(SPUF(Hash(\mathbf{c})), \mathbf{p}), Hash(\mathbf{c}))$
$\mathbf{r}$
Abort if $\mathbf{r} \neq \mathbf{r}'$

Fig. 9. Authentication with controlled PUFs. We note that they were proposed in a wider context than CRP-based token authentication only.

Incomplete Specification (#1) The authors do not suggest a method to expand the response space of the strong PUF. If the PUF is used in feedback mode, the expansion should be performed multiple times.

Suboptimal (#6) The proposal seems to be inferior to reference II-A in terms of efficiency. Or stated otherwise: inferior to their own concept of *physically obfuscated keys*. First, server storage requirements scale linearly with the number of authentications, in contrast to constant-size. Second, there seems to be overprotection against PUF modeling attacks: the preceding hash as well as the feedback mechanism could be omitted. Nevertheless, the proposal might be superior in terms of physical security. As the authors point out, PUF-based key generation relies on the secrecy of a relatively limited number of bits, i.e., the key and/or the associated PUF response. With controlled PUFs, there is no condensed secret anymore. For impersonation purposes, one would have to physically extract sufficient CRPs so as to enable a subsequent modeling attack. As for reference II-A, this includes physical attacks on the PUF as well as *Rep* and the hash front-end. Depending on the modeling resistance of the strong PUF, this corresponds to an increased number of executions of a certain physical attack, not to be confused with stopping physical attacks though. In [3], controlled arbiter PUFs were defeated with such a hybrid physical-modeling attack.

5.3 Bolotnyy et al. (March 2008)

The protocol of Bolotnyy et al. [4] is represented by Figures 7(c) and 10. PUFs are assumed to be robust against modeling. Majority voting is suggested as an error-correction mechanism, hereby evaluating the

PUF multiple times to approach the most likely response value. The authors claim token privacy under the assumption of an eavesdropping attacker.

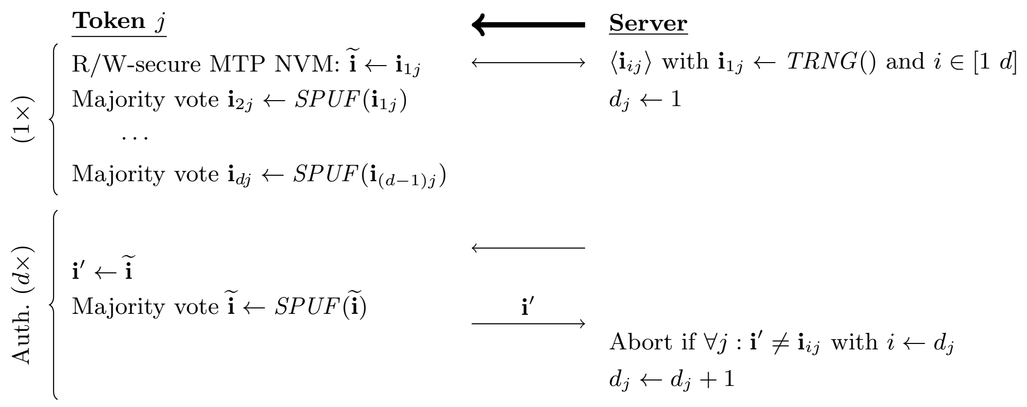

Token j
Server
(1×)
R/W-secure MTP NVM: $\widetilde{\mathbf{i}} \leftarrow \mathbf{i}_{1j}$
$\langle \mathbf{i}_{ij} \rangle$ with $\mathbf{i}_{1j} \leftarrow TRNG()$ and $i \in [1\ d]$
Majority vote $\mathbf{i}_{2j} \leftarrow SPUF(\mathbf{i}_{1j})$
$d_j \leftarrow 1$
...
Majority vote $\mathbf{i}_{dj} \leftarrow SPUF(\mathbf{i}_{(d-1)j})$
Auth. (d×)
$\mathbf{i}' \leftarrow \widetilde{\mathbf{i}}$
Majority vote $\widetilde{\mathbf{i}} \leftarrow SPUF(\widetilde{\mathbf{i}})$
$\mathbf{i}'$
Abort if $\forall j : \mathbf{i}' \neq \mathbf{i}_{ij}$ with $i \leftarrow d_j$
$d_j \leftarrow d_j + 1$

Fig. 10. Authentication protocol of Bolotnyy et al.

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.

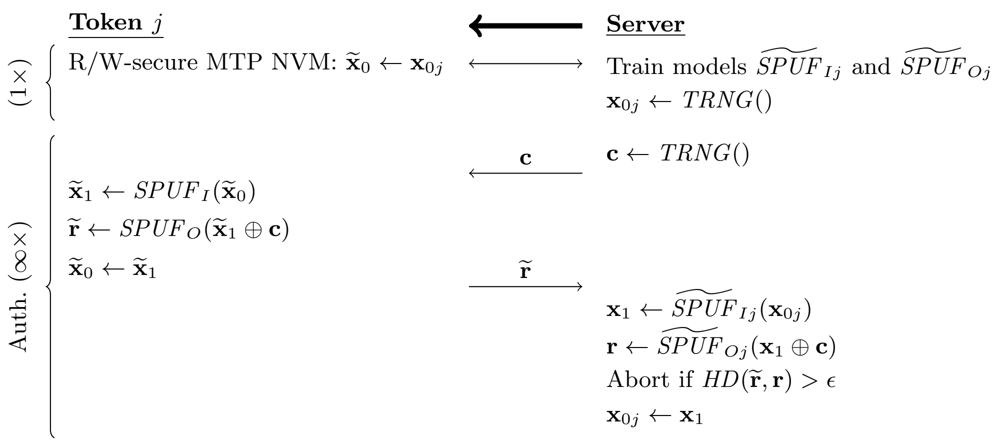
Non-Functional due to PUF Noisiness (#3) PUF noisiness has not been taken into account properly, making the protocol non-functional. Under the assumption of a uniform bit error rate, one could make the failure rate arbitrarily small. However, in practice, bit error rates are highly non-uniform, so a 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.

5.4 Öztürk et al. (March 2008)

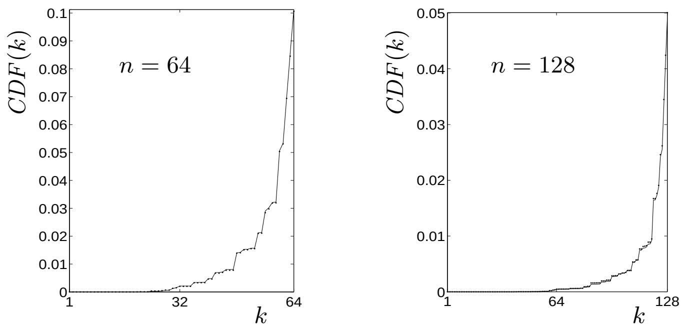
Öztürk et al. [44] use a system of two PUFs, as shown in Figure 7(d). Figure 11 represents the corresponding protocol. During enrollment, the server has both read and write access to $\widetilde{\mathbf{x}}_0$, via a one-time interface. This allows to construct models for either PUF separately, followed by an initialization of $\widetilde{\mathbf{x}}_0$. The outer PUF equips the token with a challenge-response mechanism. Its modeling by an attacker is prevented by XORing an internal secret $\widetilde{\mathbf{x}}_0$ within the challenge path. The inner PUF is employed to update $\widetilde{\mathbf{x}}_0$ continuously. The noisiness of the inner PUF makes synchronization non-trivial. At most one bit of $\widetilde{\mathbf{x}}_1$ is assumed to be affected by noise, in the seldom case of occurrence. The authors propose a simple recovery procedure at the server side: bits of $\mathbf{x}_1$ are successively flipped until the authentication succeeds. The protocol is instantiated with arbiter PUFs.



Token j
Server
(1×)
R/W-secure MTP NVM: $\widetilde{\mathbf{x}}_0 \leftarrow \mathbf{x}_{0j}$
Train models $\widetilde{SPUF}_{Ij}$ and $\widetilde{SPUF}_{Oj}$
$\mathbf{x}_{0j} \leftarrow TRNG()$
Auth. (∞×)
$\mathbf{c} \leftarrow TRNG()$
$\mathbf{c}$
$\widetilde{\mathbf{x}}_1 \leftarrow SPUF_I(\widetilde{\mathbf{x}}_0)$
$\widetilde{\mathbf{r}} \leftarrow SPUF_O(\widetilde{\mathbf{x}}_1 \oplus \mathbf{c})$
$\widetilde{\mathbf{x}}_0 \leftarrow \widetilde{\mathbf{x}}_1$
$\widetilde{\mathbf{r}}$
$\mathbf{x}_1 \leftarrow \widetilde{SPUF}_{Ij}(\mathbf{x}_{0j})$
$\mathbf{r} \leftarrow \widetilde{SPUF}_{Oj}(\mathbf{x}_1 \oplus \mathbf{c})$
Abort if $HD(\widetilde{\mathbf{r}}, \mathbf{r}) > \epsilon$
$\mathbf{x}_{0j} \leftarrow \mathbf{x}_1$

Fig. 11. Authentication protocol of Öztürk et al.

Incomplete Specification (#1) The authors use a repeated permutation to expand the response space of the inner PUF: $\forall i : \widetilde{\mathbf{x}}_1(i) \leftarrow SPUF_I(\pi^i(\widetilde{\mathbf{x}}_0))$. The permutation is not further specified but should be chosen carefully. Repetition is bound to occur, as determined by the order k: $\pi^i(\widetilde{\mathbf{x}}_0) = \pi^{i+k}(\widetilde{\mathbf{x}}_0)$. This could result in $\widetilde{\mathbf{x}}_1$ having identical bits, opposing its presumed uniformity. Figure 12 demonstrates that a significant fraction of randomly chosen permutations has $k < n$. Furthermore, the authors do not acknowledge the need to expand the response space of the outer PUF.



$$PDF(k) = \sum_{d_1|k} \mu\left(\frac{k}{d_1}\right) [x^n] \sum_{i=1}^{\infty} \frac{\left(\sum_{d_2|d_1} \frac{x^{d_2}}{d_2}\right)^i}{i!}$$

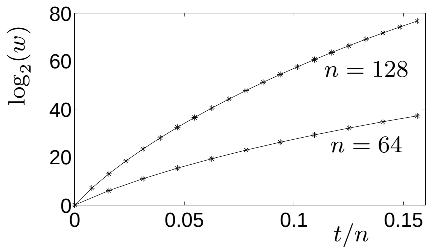
with μ the Möbius function

and $[x^n]$ the coefficient of x^n

Fig. 12. Probability distribution of the order k of a randomly chosen permutation, given a set with n elements. Dots represent simulation results originating from 100000 randomly chosen permutations. An exact mathematical formula [48], drawn continuously although of discrete nature, is hereby verified.

Secure NVM Reduces PUF Benefit (#2) One requires secure NVM, hereby undermining the main benefit of PUF technology. Either read or write access would enable an attacker to model the system, as during the enrollment. Furthermore, the motivation for the inner PUF is rather questionable. One might equally well opt for a PRNG. First, the outer PUF and the initialization of $\widetilde{\mathbf{x}}_0$ already induce IC-specific behavior. Second, its noisiness poses a limit on the complexity of the feedback loop, and hence the modeling resistance of the overall system.

PUF Noisiness Underestimated (#3) The synchronization effort is presented too optimisticly. In practice, the server faces a continuous synchronization effort, not necessarily limited to a single error. As Figure 13 demonstrates, correction might not even be feasible under very noisy circumstances.



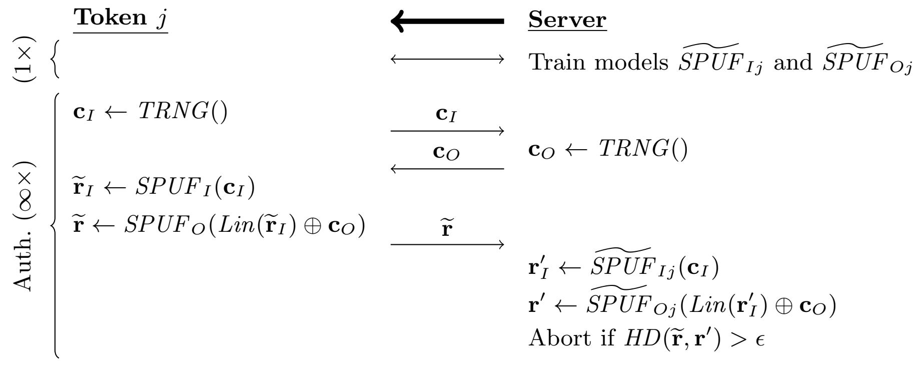
$$w = \sum_{i=0}^{t} \binom{n}{i} = \sum_{i=0}^{t} \frac{n!}{i!(n-i)!}$$

Fig. 13. The worst-case search effort w in order to correct a maximum of t a priori unknown errors in a vector of length n.

Denial-of-Service (#8) There is a simple denial-of-service attack. An attacker only needs to send a single $\mathbf{c}$ in order to desynchronize token and server. Either blocking or modifying a returned response $\widetilde{\mathbf{r}}$ has the same effect.

5.5 Hammouri et al. (December 2008)

Similar to Öztürk et al., Hammouri et al. [18] employ again two strong PUFs, as shown in Figure 7(e). As before, both PUFs are modeled during enrollment via an auxiliary one-time interface. The outer PUF is an arbiter PUF. An XOR network Lin largely compensates its non-linearity. The system $\widetilde{\mathbf{r}} \leftarrow SPUF_O(Lin(\widetilde{\mathbf{r}}_I))$ is therefore not bothered by error amplification. The inner PUF is a custom architecture based on the arbiter PUF. We refer to Appendices A and B of our CHES 2014 manuscript, which extends former high-level description with PUF internals. Figure 14 represents the authentication protocol.



Token j
Server
$(1\times)$
Train models $\widetilde{SPUF}_{Ij}$ and $\widetilde{SPUF}_{Oj}$
Auth. $(\infty\times)$
$\mathbf{c}_I \leftarrow TRNG()$
$\mathbf{c}_I$
$\mathbf{c}_O$
$\mathbf{c}_O \leftarrow TRNG()$
$\widetilde{\mathbf{r}}_I \leftarrow SPUF_I(\mathbf{c}_I)$
$\widetilde{\mathbf{r}} \leftarrow SPUF_O(Lin(\widetilde{\mathbf{r}}_I) \oplus \mathbf{c}_O)$
$\widetilde{\mathbf{r}}$
$\mathbf{r}'_I \leftarrow \widetilde{SPUF}_{Ij}(\mathbf{c}_I)$
$\mathbf{r}' \leftarrow \widetilde{SPUF}_{Oj}(Lin(\mathbf{r}'_I) \oplus \mathbf{c}_O)$
Abort if $HD(\widetilde{\mathbf{r}}, \mathbf{r}') > \epsilon$

Fig. 14. Authentication protocol of Hammouri et al.

Non-Functional due to PUF Noisiness (#3,#7) PUF noisiness has not been taken into account properly, making the protocol non-functional. Minimization of error amplification does not hold for the system $\widetilde{\mathbf{r}} \leftarrow SPUF_O(Lin(\widetilde{\mathbf{r}}_I) \oplus \mathbf{c_O})$, in the general case that $\mathbf{c_O} \neq \mathbf{0}$. Persistent token rejection is to be expected. This could have been avoided by implementing the system $\widetilde{\mathbf{r}} \leftarrow SPUF_O(Lin(\widetilde{\mathbf{r}}_I \oplus \mathbf{c_O}))$ instead. Nevertheless, two fundamental issues would remain, associated to the need for Lin. First, it degrades the modeling resistance of the overall system. Second, it makes the proposal non-generic.

Enrollment Overcomplicated (#7) Reading out $\widetilde{\mathbf{r}}_I$ via a one-time interface would have made the enrollment easy. This would allow to model both PUFs separately. However, one designed a rather complicated procedure to model the inner PUF via the response of the outer PUF. Fixing $\mathbf{c_O} = \mathbf{0}$, one obtains a system $\widetilde{\mathbf{r}} \leftarrow SPUF_O(Lin(\widetilde{\mathbf{r}}_I))$, which is not bothered by error amplification. Via a one-time interface, one modifies the response expansion method of the inner PUF: bits corresponding to the same challenge $\mathbf{c_I}$ are concatenated. Therefore, $\widetilde{\mathbf{r}}_I$ equals either $\mathbf{0}$ or $\overline{\mathbf{0}}$, apart from potential noisiness. Both cases can be distinguished, as the outer arbiter PUF is extremely likely to produce different outputs for these vectors. Unfortunately, this is not the case for the chosen expansion method of the latter: the server sends a list of challenges $\mathbf{c_O}$. For a significant fraction of the fabricated tokens, the output bit $\widetilde{r}$ might be in a noisy state, leading to yield issues.

5.6 Kulseng et al. (March 2010)

The proposal of Kulseng et al. [33] provides mutual authentication. Figures 7(f) and 15 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.

$$
\begin{array}{l l c l}
 & \underline{\textbf{Token } j} & \Longleftrightarrow & \underline{\textbf{Server}} \\
\left.\begin{array}{c} \\ (1\times) \\ \\ \end{array}\right\{ & \begin{array}{l} \text{Insecure MTP NVM: } \mathbf{i} \leftarrow \mathbf{i}_j \\ \text{R/W-secure MTP NVM: } \widetilde{\mathbf{x}}_{-1}, \widetilde{\mathbf{x}}_0 \leftarrow \mathbf{x}_{0j} \\ \text{R/W-secure OTP NVM: } \mathbf{y} \leftarrow \mathbf{y}_j \\ \mathbf{x}_{1j} \leftarrow \mathit{SPUF}(\widetilde{\mathbf{x}}_{0j}) \end{array} & \begin{array}{c} \longleftarrow \\ \longleftarrow \\ \longleftarrow \\ \longrightarrow \end{array} & \begin{array}{l} \mathbf{i}_j \leftarrow \mathit{TRNG}() \\ \mathbf{x}_{0j} \leftarrow \mathit{TRNG}() \\ \mathbf{y}_j \leftarrow \mathit{TRNG}() \\ \mathbf{x}_{1j} \end{array} \\
\text{Auth. } (\infty\times) & & \longleftarrow & \\
 & & \xrightarrow{\mathbf{i}} & \\
 & & & \text{Abort if } \mathbf{i} \neq \mathbf{i}_j \text{ or } \mathbf{i} \neq \mathit{PRNG}(\mathbf{i}_j \oplus \mathbf{x}_{0j}) \\
 & & \xleftarrow{\mathbf{b}} & \mathbf{b} \leftarrow \mathbf{y}_j \oplus \mathbf{x}_{0j} \\
 & \text{If } \mathbf{b} = \mathbf{y} \oplus \widetilde{\mathbf{x}}_0 & & \\
 & \quad \widetilde{\mathbf{x}}_1 \leftarrow \mathit{SPUF}(\widetilde{\mathbf{x}}_0) & & \\
 & \quad \widetilde{\mathbf{w}}_1 \leftarrow \mathit{PRNG}(\widetilde{\mathbf{x}}_0) & & \\
 & \quad \mathbf{i} \leftarrow \mathit{PRNG}(\mathbf{i} \oplus \widetilde{\mathbf{x}}_0) & & \\
 & \quad \widetilde{\mathbf{x}}_{-1} \leftarrow \widetilde{\mathbf{x}}_0 & & \\
 & \quad \widetilde{\mathbf{x}}_0 \leftarrow \widetilde{\mathbf{x}}_1 & & \\
 & \text{Else if } \mathbf{b} = \mathbf{y} \oplus \widetilde{\mathbf{x}}_{-1} & & \\
 & \quad \widetilde{\mathbf{x}}_1 \leftarrow \mathit{SPUF}(\widetilde{\mathbf{x}}_{-1}) & & \\
 & \quad \widetilde{\mathbf{w}}_1 \leftarrow \mathit{PRNG}(\widetilde{\mathbf{x}}_{-1}) & & \\
 & \text{Else Abort} & & \\
 & \widetilde{\mathbf{a}}_1 \leftarrow \widetilde{\mathbf{x}}_1 \oplus \widetilde{\mathbf{w}}_1 & & \\
 & \widetilde{\mathbf{x}}_2 \leftarrow \mathit{SPUF}(\widetilde{\mathbf{x}}_1) & & \\
 & \widetilde{\mathbf{w}}_2 \leftarrow \mathit{PRNG}(\mathbf{w}_1) & & \\
 & \widetilde{\mathbf{a}}_2 \leftarrow \widetilde{\mathbf{x}}_2 \oplus \widetilde{\mathbf{w}}_2 & \xrightarrow{\widetilde{\mathbf{a}}_1, \widetilde{\mathbf{a}}_2} & \\
 & & & \mathbf{w}_1 \leftarrow \mathit{PRNG}(\mathbf{x}_{0j}) \\
 & & & \text{Abort if } \mathbf{x}_{1j} \neq \widetilde{\mathbf{a}}_1 \oplus \mathbf{w}_1 \\
 & & & \mathbf{i}_j \leftarrow \mathit{PRNG}(\mathbf{i}_j \oplus \mathbf{x}_{0j}) \\
 & & & \mathbf{x}_{0j} \leftarrow \mathbf{x}_{1j} \\
 & & & \mathbf{w}_2 \leftarrow \mathit{PRNG}(\mathbf{w}_1) \\
 & & & \mathbf{x}_{1j} \leftarrow \widetilde{\mathbf{a}}_2 \oplus \mathbf{w}_2
\end{array}
$$

Fig. 15. 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 [26] 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 [26], 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 [26], 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 $\mathbf{w}_1^{(1)} = \mathbf{i}^{(2)} \oplus LFSR(\mathbf{i}^{(1)})$ and subsequently also other secret variables.

5.7 Sadeghi et al. (October 2010)

The protocol of Sadeghi et al. [53] is represented by Figures 2(f) and 16. 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.

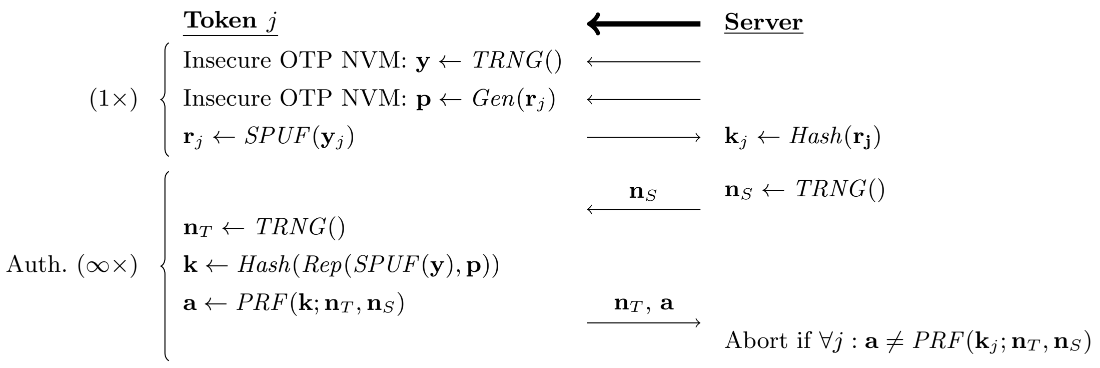

Token j
Server
$(1\times)$
Insecure OTP NVM: $\mathbf{y} \leftarrow TRNG()$
Insecure OTP NVM: $\mathbf{p} \leftarrow Gen(\mathbf{r}_j)$
$\mathbf{r}_j \leftarrow SPUF(\mathbf{y}_j)$
$\mathbf{k}_j \leftarrow Hash(\mathbf{r_j})$
Auth. $(\infty\times)$
$\mathbf{n}_S$
$\mathbf{n}_S \leftarrow TRNG()$
$\mathbf{n}_T \leftarrow TRNG()$
$\mathbf{k} \leftarrow Hash(Rep(SPUF(\mathbf{y}), \mathbf{p}))$
$\mathbf{a} \leftarrow PRF(\mathbf{k}; \mathbf{n}_T, \mathbf{n}_S)$
$\mathbf{n}_T$, $\mathbf{a}$
Abort if $\forall j : \mathbf{a} \neq PRF(\mathbf{k}_j; \mathbf{n}_T, \mathbf{n}_S)$

Fig. 16. 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.

5.8 Logically Reconfigurable PUF (September 2011)

Logically reconfigurable PUFs [28, 29] were proposed to make tokens recyclable in a privacy-preserving manner. This in order to reduce the amount of electronic waste and its associated disposal costs. Figure 17 represents the authentication protocol. The server collects CRPs which are valid for the current state $\mathbf{x}$ only. Read access to $\mathbf{x}$ is allowed, as shown in Figure 7(g), although not required. There is no direct write access, although one can perform an update: $\mathbf{x} \leftarrow Hash(\mathbf{x})$. The server then needs to collect new CRPs. An attacker can only track tokens in between state updates. We note that this corresponds to a rather limited form of privacy only: other proposals claim to prohibit tracking across authentications.

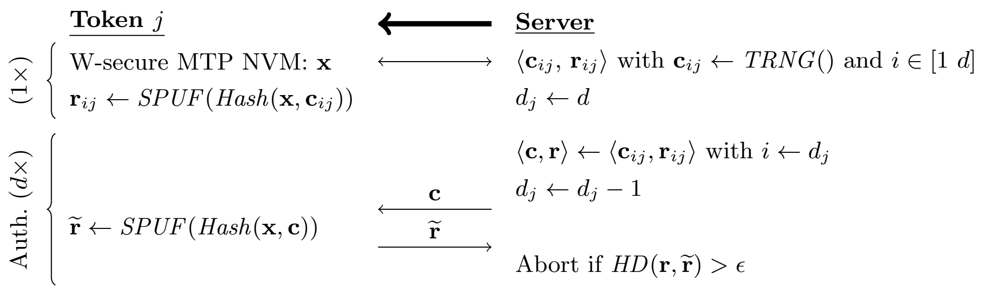

Token j
Server
(1×)
W-secure MTP NVM: $\mathbf{x}$
$\mathbf{r}_{ij} \leftarrow SPUF(Hash(\mathbf{x}, \mathbf{c}_{ij}))$
$\langle \mathbf{c}_{ij}, \mathbf{r}_{ij} \rangle$ with $\mathbf{c}_{ij} \leftarrow TRNG()$ and $i \in [1\ d]$
$d_j \leftarrow d$
Auth. (d×)
$\langle \mathbf{c}, \mathbf{r} \rangle \leftarrow \langle \mathbf{c}_{ij}, \mathbf{r}_{ij} \rangle$ with $i \leftarrow d_j$
$d_j \leftarrow d_j - 1$
$\mathbf{c}$
$\widetilde{\mathbf{r}} \leftarrow SPUF(Hash(\mathbf{x}, \mathbf{c}))$
$\widetilde{\mathbf{r}}$
Abort if $HD(\mathbf{r}, \widetilde{\mathbf{r}}) > \epsilon$

Fig. 17. Authentication protocol for logically reconfigurable PUFs.

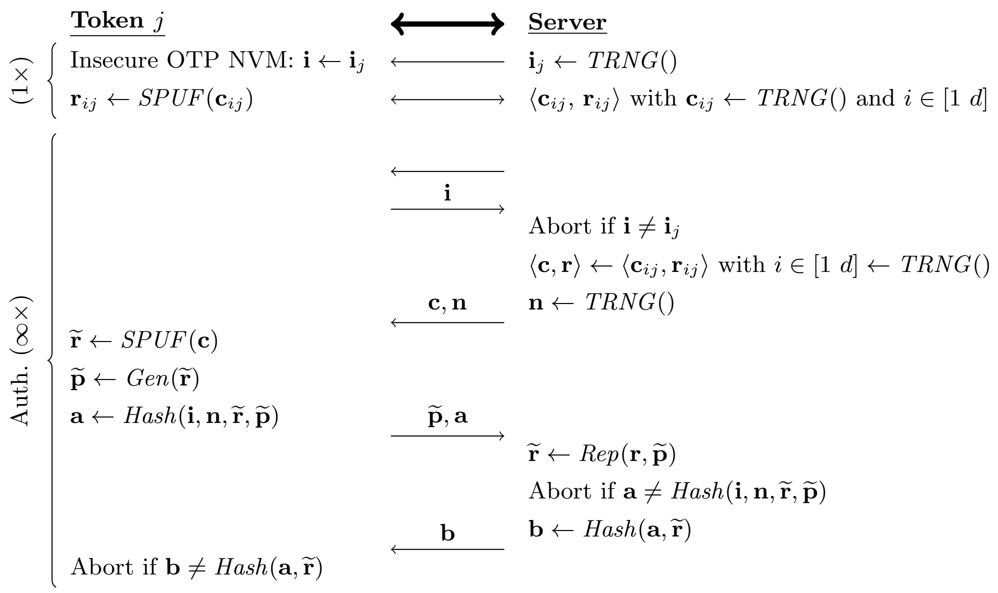
NVM Undermines PUF Benefit (#2) The proposal requires write-secure NVM. This partly undermines the main benefit of PUF technology: improved physical security. Together with the hash, the need for MTP NVM also degrades the economical gain with respect to basic authentication. Disposable but low-cost tokens are transformed to recyclable but expensive tokens.

Modeling Attacks (#4) The proposal does not aim to prevent PUF modeling attacks. Therefore, the practical value of the proposal is rather limited: the protocol cannot be instantiated due to lack of an appropriate strong PUF. An attacker cannot fully control the PUF input, but this is insufficient as defense: most modeling attacks do not rely on a chosen-challenge advantage.

Denial-of-Service (#8) There is a simple denial-of-service attack. An attacker can update $\mathbf{x}$ and hence invalidate the CRP database of the server. The proposal does not describe a user authentication mechanism for the reconfiguration, which is required to stop this attack.

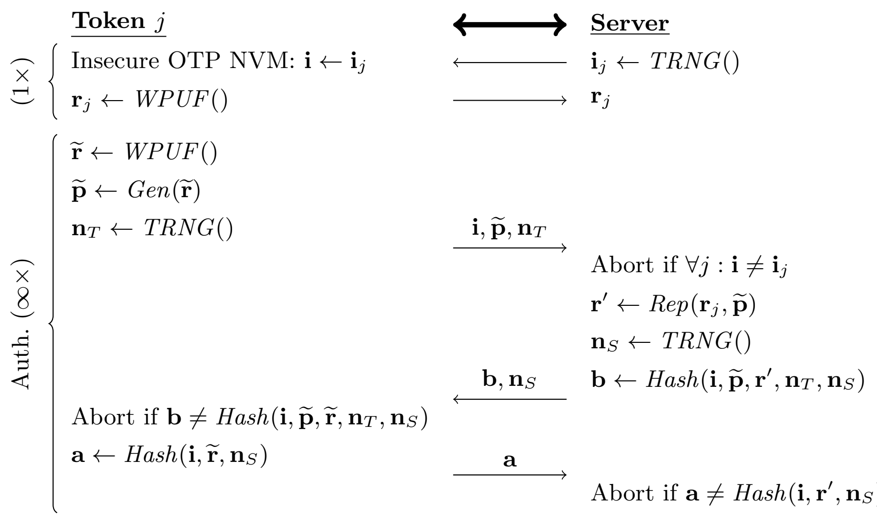
5.9 Reverse Fuzzy Extractors: Predecessor (February 2012) and Successor (August 2012)

The reverse FE proposal [59] provides mutual authentication. The term 'reverse' highlights that *Gen* and not *Rep* is implemented on every token, as shown in Figure 7(h). As such, one does benefit of the lightweight potential of *Gen*. Figure 18 represents the original protocol. The authors raise the concern of repeated helper data exposure: an attacker might collect helper data $\{\widetilde{\mathbf{p}}^{(1)}, \widetilde{\mathbf{p}}^{(2)}, \ldots\}$ for the same challenge $\mathbf{c}$. Therefore, they recommend the syndrome construction, as there is provably no additional leakage with respect to the traditional $n-k$ entropy loss. The proof-of-concept implementation makes use of an arbiter PUF. One author later proposed a modified protocol [37], as represented by Figures 2(g) and 19. A reversal of authentication checks is stated to be the main difference, although there seem to be other fundamental changes as clarified hereafter.



Token j
Server
(1×)
Insecure OTP NVM: $\mathbf{i} \leftarrow \mathbf{i}_j$
$\mathbf{i}_j \leftarrow \mathit{TRNG}()$
$\mathbf{r}_{ij} \leftarrow \mathit{SPUF}(\mathbf{c}_{ij})$
$\langle \mathbf{c}_{ij}, \mathbf{r}_{ij} \rangle$ with $\mathbf{c}_{ij} \leftarrow \mathit{TRNG}()$ and $i \in [1\ d]$
Auth. (∞×)
$\mathbf{i}$
Abort if $\mathbf{i} \neq \mathbf{i}_j$
$\langle \mathbf{c}, \mathbf{r} \rangle \leftarrow \langle \mathbf{c}_{ij}, \mathbf{r}_{ij} \rangle$ with $i \in [1\ d] \leftarrow \mathit{TRNG}()$
$\mathbf{c}, \mathbf{n}$
$\mathbf{n} \leftarrow \mathit{TRNG}()$
$\widetilde{\mathbf{r}} \leftarrow \mathit{SPUF}(\mathbf{c})$
$\widetilde{\mathbf{p}} \leftarrow \mathit{Gen}(\widetilde{\mathbf{r}})$
$\mathbf{a} \leftarrow \mathit{Hash}(\mathbf{i}, \mathbf{n}, \widetilde{\mathbf{r}}, \widetilde{\mathbf{p}})$
$\widetilde{\mathbf{p}}, \mathbf{a}$
$\widetilde{\mathbf{r}} \leftarrow \mathit{Rep}(\mathbf{r}, \widetilde{\mathbf{p}})$
Abort if $\mathbf{a} \neq \mathit{Hash}(\mathbf{i}, \mathbf{n}, \widetilde{\mathbf{r}}, \widetilde{\mathbf{p}})$
$\mathbf{b}$
$\mathbf{b} \leftarrow \mathit{Hash}(\mathbf{a}, \widetilde{\mathbf{r}})$
Abort if $\mathbf{b} \neq \mathit{Hash}(\mathbf{a}, \widetilde{\mathbf{r}})$

Fig. 18. Original reverse FE protocol.



Token j
Server
(1×)
Insecure OTP NVM: $\mathbf{i} \leftarrow \mathbf{i}_j$
$\mathbf{i}_j \leftarrow \mathit{TRNG}()$
$\mathbf{r}_j \leftarrow \mathit{WPUF}()$
$\mathbf{r}_j$
Auth. (∞×)
$\widetilde{\mathbf{r}} \leftarrow \mathit{WPUF}()$
$\widetilde{\mathbf{p}} \leftarrow \mathit{Gen}(\widetilde{\mathbf{r}})$
$\mathbf{n}_T \leftarrow \mathit{TRNG}()$
$\mathbf{i}, \widetilde{\mathbf{p}}, \mathbf{n}_T$
Abort if $\forall j : \mathbf{i} \neq \mathbf{i}_j$
$\mathbf{r}' \leftarrow \mathit{Rep}(\mathbf{r}_j, \widetilde{\mathbf{p}})$
$\mathbf{n}_S \leftarrow \mathit{TRNG}()$
$\mathbf{b}, \mathbf{n}_S$
$\mathbf{b} \leftarrow \mathit{Hash}(\mathbf{i}, \widetilde{\mathbf{p}}, \mathbf{r}', \mathbf{n}_T, \mathbf{n}_S)$
Abort if $\mathbf{b} \neq \mathit{Hash}(\mathbf{i}, \widetilde{\mathbf{p}}, \widetilde{\mathbf{r}}, \mathbf{n}_T, \mathbf{n}_S)$
$\mathbf{a} \leftarrow \mathit{Hash}(\mathbf{i}, \widetilde{\mathbf{r}}, \mathbf{n}_S)$
$\mathbf{a}$
Abort if $\mathbf{a} \neq \mathit{Hash}(\mathbf{i}, \mathbf{r}', \mathbf{n}_S)$

Fig. 19. Modified reverse FE protocol.

Suboptimal (#6) Even with $d = 1$, the protocol still allows for an unlimited number of authentications. Token impersonation via replay has already been prevented by the use of nonce $\mathbf{n}$. This observation could result in numerous protocol simplifications, as has been acknowledged in [59]. The authors do not state clearly though that there would a simplification for the PUF as well: the enormous input-output space is no more required, even a weak PUF could be employed. Despite all the former, one still promotes the use of $d > 1$, arguing that it offers an increased side channel resistance. For impersonation purposes, we argue that one only would have to physically extract sufficient CRPs so as to enable a subsequent modeling attack. This includes physical attacks on the PUF as well as the front-ends of *Gen* and the hash. Depending on the modeling resistance of the strong PUF, this corresponds to an increased number of executions of a certain physical attack, not to be confused with stopping physical attacks though. Against arbiter PUFs, as used in the proof-of-concept implementation, former hybrid physical-modeling attack might perform well [3]. Therefore, the suggested countermeasure might not outweigh the missed advantages and there might be numerous more rewarding countermeasures. The modified protocol proposal [37] does not discuss the foregoing matter, although it uses $d = 1$, hereby suggesting the use of a weak SRAM PUF.

PUF Noisiness Lower Bound (#7) Non-determinism is essential to avoid replay attacks. For tokens, one does exploit the PUF noisiness for this purpose, hereby avoiding the need for a TRNG. A lower bound for the PUF noisiness is imposed, in order to avoid server impersonation. However, this seems to be a very delicate balancing exercise in practice, opposing more conventional design efforts to stabilize PUFs. The need for environmental robustness magnifies this opposition. An attacker might immerse the IC in an extremely stable environment, hereby minimizing the PUF error rate. For genuine use however, one might have to provide correctness despite large perturbations, corresponding to the maximum error rate. The modified protocol proposal [37] does not discuss the foregoing contradiction, although its use of a token TRNG provides a solution.

PRNG Exploitation (#8,#5) The proof-of-concept implementation uses a PRNG to expand the 1-bit response of an arbiter PUF: $\widetilde{\mathbf{r}} \leftarrow \mathit{SPUF}(\mathit{LFSR}(\mathbf{c}))$. Due to code size limitations, $\widetilde{\mathbf{r}}$ is subdivided in x non-overlapping sections: $\widetilde{\mathbf{r}} = \left(\widetilde{\mathbf{r}}_1\ \widetilde{\mathbf{r}}_2\ \ldots\ \widetilde{\mathbf{r}}_x\right)$, with each section having length n. A corresponding helper data list $\widetilde{\mathbf{p}} = \left(\widetilde{\mathbf{p}}_1\ \widetilde{\mathbf{p}}_2\ \ldots\ \widetilde{\mathbf{p}}_x\right)$, with each section having length $n-k$, is transferred to the server. The authors use $x = 7$ and $k = 21$, aiming at a security level of $128 < x \cdot k$ bit. As a side note, the proposal does not provide an explicit warning to refuse fixed points of the LFSR, e.g., $\mathbf{c} = \mathbf{0}$. This would have been appropriate in order to avoid a trivial server impersonation attack. Fixed points will result in either $\widetilde{\mathbf{r}} = \mathbf{0}$ or $\widetilde{\mathbf{r}} = \overline{\mathbf{0}}$, assuming stability for one particular response bit. An attacker could hence guess $\mathbf{b}$ with success probability 1/2. A PRNG-related token impersonation threat, which is far less obvious, is described next.

For simplicity, consider the unrealistic but usually desired case of a perfectly deterministic PUF, at least for now. Also consider an arbitrary response section, e.g., $\mathbf{r}_1$. Helper data leakage can be understood via an underdetermined system of linear equations: $\mathbf{H} \cdot {\mathbf{r}_1}^T = {\mathbf{p}_1}^T$, having $n - k$ equations for n unknowns. Consider a challenge $\mathbf{c}^{(1)}$ leading to an response $\mathbf{r}_1^{(1)} = \left(r_1\ r_2\ \ldots\ r_n\right)$. One could easily construct a challenge $\mathbf{c}^{(2)}$ leading to a response $\mathbf{r}_1^{(2)} = \left(r_2\ r_3\ \ldots\ r_{n+1}\right)$, given the use of an LFSR. Repeating the former mechanism, we can construct challenges $\mathbf{c}^{(3)}$, $\mathbf{c}^{(4)}$, ..., $\mathbf{c}^{(q)}$, with $\mathbf{r}^{(q)} = \left(r_q\ r_{q+1}\ \ldots\ r_{q+n-1}\right)$. An attacker collects all data in a single system of equations:

$$\mathbf{A} \cdot \left(r_1\ r_2\ \ldots\ r_{q+n-1}\right)^T = \begin{pmatrix} \left(\mathbf{p}_1^{(1)}\right)^T \\ \left(\mathbf{p}_1^{(2)}\right)^T \\ \vdots \\ \left(\mathbf{p}_1^{(q)}\right)^T \end{pmatrix} \quad \text{with } \mathbf{A} = \begin{pmatrix} \mathbf{H} & \mathbf{0}^T & \ldots & \mathbf{0}^T \\ \mathbf{0}^T & \mathbf{H} & \cdots & \mathbf{0}^T \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}^T & \mathbf{0}^T & \cdots & \mathbf{H} \end{pmatrix}.$$

Even for small q, an attacker easily obtains more equations than unknowns. Equation dependencies still have to be considered though. We performed experiments where $\mathbf{A}$ is transformed to reduced row echelon form (*rref*). A distinction between cyclic and non-cyclic codes seems to be crucial, as illustrated in Figure 20. In the latter case, an attacker is able to solve the system. The persistence of few unknowns is only a minor inconvenience. In the former case, the number of unknowns remains k, independent of q. However, for large q, a divide-and-conquer machine learning attack can be performed instead, exploiting the introduction of sections. Arbiter PUFs can be modeled with only a few thousand CRPs, so consider, e.g., $q = 10000$, including both training and verification data. The correct combination of unknowns (only $2^k = 2^{21}$ possibilities) would result in the observable event of a high modeling accuracy.

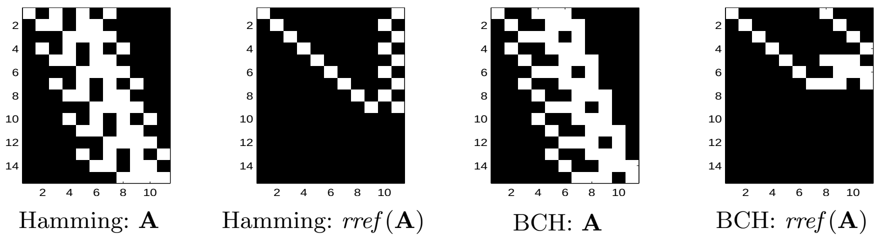


Fig. 20. PRNG exploitation for the reverse fuzzy extractor: illustration for non-cyclic Hamming and cyclic BCH codes, with $[n = 7,\ k = 4,\ t = 1]$ and $q = 5$. Black and white represent 0 and 1 respectively.

We now consider a PUF which is bothered by noise. To minimize this inconvenience, an attacker could ensure the IC's environment to be very stable. Code parameters, which are normally chosen in order to maintain robustness against a variety of environmental perturbations (supply voltage, outside temperature, etc.), become relatively relaxed then. Subsequently, we consider a transformation $\mathbf{H}^{\star} = \mathbf{T} \cdot \mathbf{H}$, which selects linear combinations of the rows of $\mathbf{H}$. The transformation is chosen so that the rows of $\mathbf{H}^{\star}$ do have a low Hamming weight, making it feasible to incorporate a limited set of stable bits only. One could develop a variety of algorithms in order to find a suitable transformation. Recent follow-up work in [2] experimentally confirms the validity of the PRNG issue, i.e., in the presence of noise. We also stress that former attacks do not apply to the modified protocol proposal [37], as there is no PRNG.

5.10 Converse Authentication: Predecessor (March 2012) and Successor (June 2012)

Figures 7(i) and 21 represent the latest version of the converse authentication protocol [30]. The authentication is one-way, in a less conventional setting where tokens verify the server. The difference vector (XOR) between two responses is denoted as $\mathbf{a}$. The restriction $\mathbf{a} \neq \mathbf{0}$ prevents an attacker from impersonating the server, choosing $\mathbf{c}_u = \mathbf{c}_v$ and $\mathbf{p}_u = \mathbf{p}_v$. Optionally, one can extend the protocol with the establishment of a session key $\mathbf{k} \leftarrow \mathit{Hash}(\mathbf{r}_u, \mathbf{r}_v)$. The attacker capabilities are restricted: invasive analysis of the prover IC is assumed to be impossible. Furthermore, one assumes an eavesdropping attacker, trying to impersonate the server given a log of genuine authentication transcripts. The authors suggest the use of arbiter PUFs.

	Token j	$\longrightarrow$	**Server**
(1×)	$\mathbf{r}_{ij} \leftarrow \mathit{Hash}(\mathit{SPUF}(\mathbf{c}_{ij}))$	$\longleftrightarrow$	$\langle \mathbf{c}_{ij}, \mathbf{p}_{ij}, \mathbf{r}_{ij} \rangle$ with $\mathbf{c}_{ij} \leftarrow \mathit{TRNG}()$ and $i \in [1\ d]$
	$\mathbf{p}_{ij} \leftarrow \mathit{Gen}(\mathit{SPUF}(\mathbf{c}_{ij}))$		
Auth. (∞×)	$\mathbf{a} \neq \mathbf{0} \leftarrow \mathit{TRNG}()$	$\xrightarrow{\mathbf{a}}$	
			Find u, v such that $\mathbf{r}_{uj} \oplus \mathbf{r}_{vj} = \mathbf{a}$
			If none, choose random u, v
		$\xleftarrow{\mathbf{c}_u,\ \mathbf{p}_u,\ \mathbf{c}_v,\ \mathbf{p}_v}$	$\mathbf{c}_u \leftarrow \mathbf{c}_{uj}$, $\mathbf{p}_u \leftarrow \mathbf{p}_{uj}$, $\mathbf{c}_v \leftarrow \mathbf{c}_{vj}$ and $\mathbf{p}_v \leftarrow \mathbf{p}_{vj}$
	$\mathbf{r}_u \leftarrow \mathit{Hash}(\mathit{Rep}(\mathit{SPUF}(\mathbf{c}_u), \mathbf{p}_u))$		
	$\mathbf{r}_v \leftarrow \mathit{Hash}(\mathit{Rep}(\mathit{SPUF}(\mathbf{c}_v), \mathbf{p}_v))$		
	Abort if $\mathbf{r}_u \oplus \mathbf{r}_v \neq \mathbf{a}$		

Fig. 21. Converse authentication protocol.

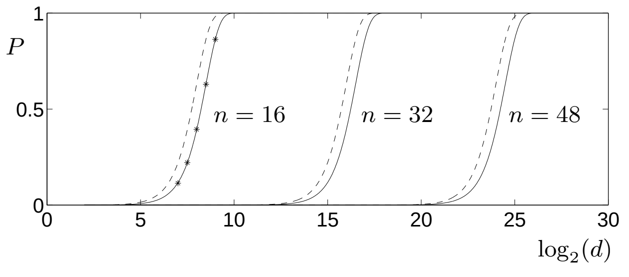
Predecessor Issues Although the protocol is in essence identical to a direct predecessor [7], it is not described in terms of adjustments. We observe six modifications. First, *Rep* is acknowledged to require helper data. Second, one introduces the restriction $\mathbf{a} \neq \mathbf{0}$. However, an explicit check could be omitted given its negligible probability of occurrence. Third, $\mathbf{a}$ is generated by a TRNG instead of a PRNG. To avoid replay, the latter construction would require secure NVM, hereby undermining the advantages of PUFs. Fourth, attacker capabilities are reduced. Fifth, the protocol is initiated by a token instead of the server. Sixth, PUF responses are reinforced by a cryptographic hash function. In its absence, we see a direct opportunity for prover impersonation in the occasional case that $HW(\mathbf{a}) < t$. Consider an arbitrary response $\mathbf{r} = Rep(SPUF(\mathbf{c}), \mathbf{p})$. For both secure sketch constructions, an attacker might be able to produce a given $\mathbf{a}$:

Code-offset construction: $\mathbf{r} \oplus \mathbf{a} = Rep(SPUF(\mathbf{c}), \mathbf{p} \oplus \mathbf{a})$.
Syndrome construction: $\mathbf{r} \oplus \mathbf{a} = Rep(SPUF(\mathbf{c}), \mathbf{p} \oplus \mathbf{a} \cdot \mathbf{H}^T)$.

Unclear and Overly Restricted Attacker Capabilities (#1) The attacker capabilities are unclear and overly restricted to be practical. First, restricting invasion would automatically extend to physical attacks in general, including side channel analysis. A real-life attacker is not inclined to obey artificial constraints. Furthermore, this restriction would greatly reduce the need for PUFs, with respect to the traditional approach of storing a key in NVM. It is also not clear whether an attacker is allowed to freely query the server, as exploited hereafter. We argue that this should be possible as well, as the protocol is initiated by a token.

Suboptimal (#6) The proposal seems to be inferior to reference II-B in terms of efficiency. First, the PUF is required to have an enormous instead of modest input-output space, which is inherently more demanding. Second, the PUF and its fuzzy extractor both need to be evaluated twice, instead of only once, in addition to extra XORing logic. Third, server storage requirements and its associated computational effort are both enormous instead of modest.

Brute-force (#8) The proposal does not scale to a comfortable security level. The server can be impersonated via brute-force, independent of who would initiate the protocol. Successful authentication relies on the availability of a given $\mathbf{a}$ within the server database. As is clear from Figure 22, token-specific CRP databases should be roughly of size $d = 2^{\frac{n}{2}+1}$. Although the memory requirements are enormous with respect to other proposals, the computational effort is even worse. An efficient method to (construct and) search within a database was not discussed and seems far from obvious. We assume an exhaustive trial and error procedure: each of the $\frac{d(d-1)}{2}$ pairwise selections has a probability of $\frac{1}{2^n-1}$ to produce the given $\mathbf{a}$. A database of size $d = 2^{25}$ is mentioned to be realistic. This would correspond to circa 2^{49} pairs and a security level $n = 48$. As tokens initiate the protocol, an attacker might exhaustively query the server and construct a personal database: $\langle \mathbf{a}_i, \mathbf{c}_{i1}, \mathbf{p}_{i1}, \mathbf{c}_{i2}, \mathbf{p}_{i2} \rangle$. Authentication will succeed, although a session key cannot be retrieved if present. If the server would initiate the protocol, there is a similar threat. Random guessing has a probability of $\frac{1}{2^n}$ to be successful. In summary, attacker and server face a similar computational effort. Even worse: the server faces this effort for every authentication, while an attacker only needs to do it once.



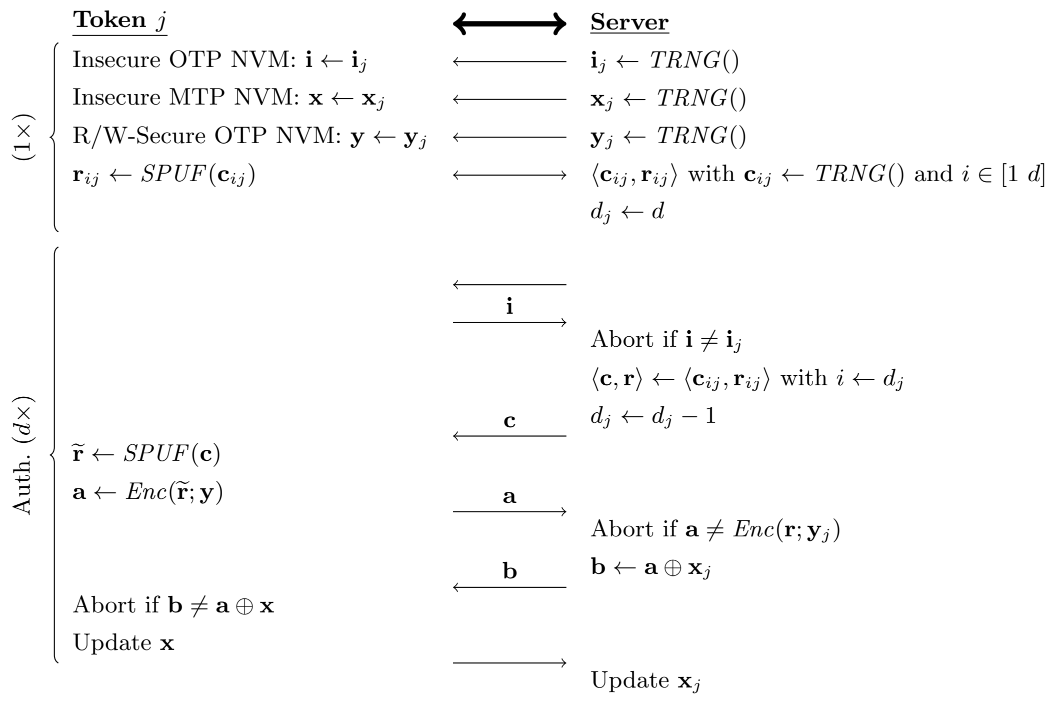
$$P = 1 - \left(1 - \frac{1}{2^n-1}\right)^{\frac{d(d-1)}{2}}$$

with $\mathbf{r}$ and $\mathbf{a} \neq \mathbf{0}$ uniformly distributed

Fig. 22. Scalability of the converse authentication protocol. We represent the probability P that a randomly chosen $\mathbf{a} \neq \mathbf{0}$ is covered by the database. An exact formula is drawn with solid lines for $n \in \{16, 32, 48\}$. We believe a similar formula in [30] to be incorrect, slightly overestimating P as clear from the dashed lines. Stars represent independent simulation results originating from 100000 authentications, verifying the correctness of our formula, only for $n = 16$ because of the runtime.

5.11 Lee et al. I (March 2012)

The first protocol of Lee et al. [35] is represented by Figures 7(j) and 23. 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.



Token j
Server
$(1\times)$
Insecure OTP NVM: $\mathbf{i} \leftarrow \mathbf{i}_j$
Insecure MTP NVM: $\mathbf{x} \leftarrow \mathbf{x}_j$
R/W-Secure OTP NVM: $\mathbf{y} \leftarrow \mathbf{y}_j$
$\mathbf{r}_{ij} \leftarrow SPUF(\mathbf{c}_{ij})$
$\mathbf{i}_j \leftarrow TRNG()$
$\mathbf{x}_j \leftarrow TRNG()$
$\mathbf{y}_j \leftarrow TRNG()$
$\langle \mathbf{c}_{ij}, \mathbf{r}_{ij} \rangle$ with $\mathbf{c}_{ij} \leftarrow TRNG()$ and $i \in [1\ d]$
$d_j \leftarrow d$
Auth. $(d\times)$
$\mathbf{i}$
Abort if $\mathbf{i} \neq \mathbf{i}_j$
$\langle \mathbf{c}, \mathbf{r} \rangle \leftarrow \langle \mathbf{c}_{ij}, \mathbf{r}_{ij} \rangle$ with $i \leftarrow d_j$
$d_j \leftarrow d_j - 1$
$\mathbf{c}$
$\widetilde{\mathbf{r}} \leftarrow SPUF(\mathbf{c})$
$\mathbf{a} \leftarrow Enc(\widetilde{\mathbf{r}}; \mathbf{y})$
$\mathbf{a}$
Abort if $\mathbf{a} \neq Enc(\mathbf{r}; \mathbf{y}_j)$
$\mathbf{b} \leftarrow \mathbf{a} \oplus \mathbf{x}_j$
$\mathbf{b}$
Abort if $\mathbf{b} \neq \mathbf{a} \oplus \mathbf{x}$
Update $\mathbf{x}$
Update $\mathbf{x}_j$

Fig. 23. 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.

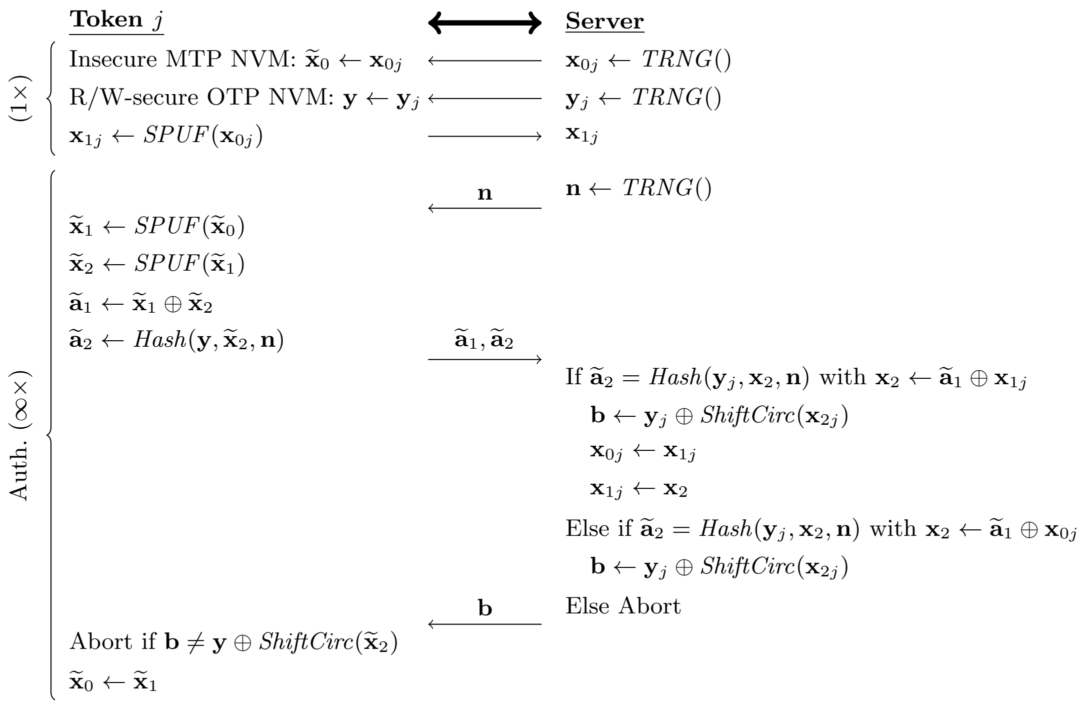
Non-Functional due to PUF Noisiness (#3) The authors mention the existence of fuzzy extractors as an error-correction mechanism. However, *Gen*/*Rep* procedures are not part of the proposed protocol and the need for helper data is not acknowledged either. We assume the complete absence of error-correction 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 [43], so one might have to consider other ciphers.

5.12 Jin et al. (April 2012)

The proposal of Jin et al. [23] seems to be inspired by [33]. The protocol is represented by Figures 7(k) and 24. 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.



Token j $\longleftrightarrow$ **Server**

(1×)

Insecure MTP NVM: $\widetilde{\mathbf{x}}_0 \leftarrow \mathbf{x}_{0j}$ $\longleftarrow$ $\mathbf{x}_{0j} \leftarrow \mathit{TRNG}()$

R/W-secure OTP NVM: $\mathbf{y} \leftarrow \mathbf{y}_j$ $\longleftarrow$ $\mathbf{y}_j \leftarrow \mathit{TRNG}()$

$\mathbf{x}_{1j} \leftarrow \mathit{SPUF}(\mathbf{x}_{0j})$ $\longrightarrow$ $\mathbf{x}_{1j}$

Auth. (∞×)

$\xleftarrow{\mathbf{n}}$ $\mathbf{n} \leftarrow \mathit{TRNG}()$

$\widetilde{\mathbf{x}}_1 \leftarrow \mathit{SPUF}(\widetilde{\mathbf{x}}_0)$

$\widetilde{\mathbf{x}}_2 \leftarrow \mathit{SPUF}(\widetilde{\mathbf{x}}_1)$

$\widetilde{\mathbf{a}}_1 \leftarrow \widetilde{\mathbf{x}}_1 \oplus \widetilde{\mathbf{x}}_2$

$\widetilde{\mathbf{a}}_2 \leftarrow \mathit{Hash}(\mathbf{y}, \widetilde{\mathbf{x}}_2, \mathbf{n})$ $\xrightarrow{\widetilde{\mathbf{a}}_1, \widetilde{\mathbf{a}}_2}$

If $\widetilde{\mathbf{a}}_2 = \mathit{Hash}(\mathbf{y}_j, \mathbf{x}_2, \mathbf{n})$ with $\mathbf{x}_2 \leftarrow \widetilde{\mathbf{a}}_1 \oplus \mathbf{x}_{1j}$

  $\mathbf{b} \leftarrow \mathbf{y}_j \oplus \mathit{ShiftCirc}(\mathbf{x}_{2j})$

  $\mathbf{x}_{0j} \leftarrow \mathbf{x}_{1j}$

  $\mathbf{x}_{1j} \leftarrow \mathbf{x}_2$

Else if $\widetilde{\mathbf{a}}_2 = \mathit{Hash}(\mathbf{y}_j, \mathbf{x}_2, \mathbf{n})$ with $\mathbf{x}_2 \leftarrow \widetilde{\mathbf{a}}_1 \oplus \mathbf{x}_{0j}$

  $\mathbf{b} \leftarrow \mathbf{y}_j \oplus \mathit{ShiftCirc}(\mathbf{x}_{2j})$

$\xleftarrow{\mathbf{b}}$ Else Abort

Abort if $\mathbf{b} \neq \mathbf{y} \oplus \mathit{ShiftCirc}(\widetilde{\mathbf{x}}_2)$

$\widetilde{\mathbf{x}}_0 \leftarrow \widetilde{\mathbf{x}}_1$

Fig. 24. 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 $\mathbf{b}^{(2)} \leftarrow \mathbf{b}^{(1)} \oplus \mathit{ShiftCirc}(\widetilde{\mathbf{a}}_1^{(2)})$.

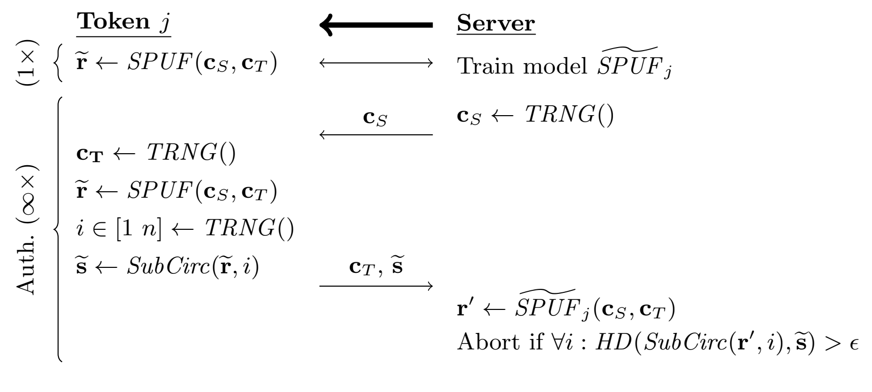
Denial-of-Service (#8) The proposal instantiates the hash computation as follows: $\widetilde{\mathbf{a}}_2 \leftarrow \mathit{Hash}(\mathbf{y}, \widetilde{\mathbf{x}}_2, \mathbf{n}) = \mathit{Hash}(\mathbf{y} \oplus \widetilde{\mathbf{x}}_2 \oplus \mathbf{n})$. This allows for desynchronization attacks: 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.

5.13 Slender PUFs: Predecessor (May 2012) and Successor (January 2014)

The original slender PUF proposal [40] includes three countermeasures against modeling, while avoiding the need for cryptographic primitives, as is clear from Figure 7(l). First, the protocol requires a strong PUF with a high resistance. A machine learning model is constructed during enrollment via auxiliary one-time interfaces. The authors opt for an XOR arbiter PUF with swapped inputs. Second, the exposure of $\widetilde{\mathbf{r}}$ is limited to random substrings $\widetilde{\mathbf{s}}$, hereby obfuscating the CRP link. The corresponding procedure *SubCirc* treats the bits in a circular manner. 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 25 represents the protocol. For completeness, we note the design to be inspired by the pattern matching key generator [46], which serves as an alternative for a fuzzy extractor.



Token j
Server
$(1\times)$
$\widetilde{\mathbf{r}} \leftarrow \mathit{SPUF}(\mathbf{c}_S, \mathbf{c}_T)$
Train model $\widetilde{\mathit{SPUF}}_j$
Auth. $(\infty\times)$
$\mathbf{c}_S$
$\mathbf{c}_S \leftarrow \mathit{TRNG}()$
$\mathbf{c}_\mathbf{T} \leftarrow \mathit{TRNG}()$
$\widetilde{\mathbf{r}} \leftarrow \mathit{SPUF}(\mathbf{c}_S, \mathbf{c}_T)$
$i \in [1\ n] \leftarrow \mathit{TRNG}()$
$\widetilde{\mathbf{s}} \leftarrow \mathit{SubCirc}(\widetilde{\mathbf{r}}, i)$
$\mathbf{c}_T, \widetilde{\mathbf{s}}$
$\mathbf{r}' \leftarrow \widetilde{\mathit{SPUF}}_j(\mathbf{c}_S, \mathbf{c}_T)$
Abort if $\forall i : \mathit{HD}(\mathit{SubCirc}(\mathbf{r}', i), \widetilde{\mathbf{s}}) > \epsilon$

Fig. 25. Slender PUF protocol.

A protocol extension has been proposed in [49]. One presents a fourth countermeasure against modeling. Substring $\widetilde{\mathbf{s}}$ is padded with random bits before transmission, again in a circular manner: $\mathit{SubCirc}(\widetilde{\mathbf{s}}_P, k) \leftarrow \widetilde{\mathbf{s}}$ with $\widetilde{\mathbf{s}}_P \in \{0,1\}^{1 \times n_P} \leftarrow \mathit{TRNG}()$ and $k \in [1\ n_P] \leftarrow \mathit{TRNG}()$. Furthermore, the optional establishment of a session key is introduced, via concatenation of secret indices i and k. A repeated execution of the protocol is required to obtain a key of sufficient length.

Exacting PUF Requirements (#7) Instantiating the protocol is difficult in practice, compared to other proposals. 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 and retrieve indices i (and k) as such. For a monolithic PUF, this would be an extremely difficult balancing exercise. For composite PUFs, such as the proposed XOR construction, it works out though. During enrollment, the arbiter chains can be modeled separately, bypassing the XOR via one-time interfaces. Specialized attacks might estimate a lower bound for the number of chains that needs to be XORed. E.g., in [2], one defeats both the original and extended protocol using simulated arbiter XOR PUFs with up to four chains, without swapping inputs across chains though. Unfortunately, it is difficult to obtain a safe estimate, as one can never exclude the possibility that an alternative attack might perform better. In addition, XORing amplifies noisiness, posing a practical upper limit on the number of chains.

There is another opposition for the extended protocol with key establishment. On one hand, cryptographic primitives which prevent modeling are avoided, in order to save resources. On the other hand, establishing a key is only useful if there is a cryptographic primitive to make use of it. We argue that the latter primitive might equally well be reused as protocol reinforcement. For completeness, we note that similar contradictions exist for the pattern matching key generator. Furthermore, the correlations threat has not been anticipated as thoroughly as in [40]. A helper data manipulation threat has not been anticipated either [9].

PRNG Exploitation (#8,#5) The protocol employs a PRNG to expand the PUF response space: $\widetilde{\mathbf{r}} \leftarrow \mathit{SPUF}(\mathit{PRNG}(\mathbf{c}_S || \mathbf{c}_T))$. However, the PRNG construction of the proof-of-concept implementation might allow for token impersonation: $\mathit{PRNG}(\mathbf{c}_S || \mathbf{c}_T) = \mathit{LFSR}(\mathbf{c}_S) \oplus \mathit{LFSR}(\mathbf{c}_T)$. We assume an identical feedback polynomial for both LFSRs, which is the most intuitive assumption[4]. A malicious token might then return $\mathbf{c_T} \leftarrow \mathbf{c_S}$, resulting in an expanded list of challenges all equal to $\mathbf{0}$. The server's PUF model outputs either $\mathbf{r}' = \mathbf{0}$ or $\mathbf{r}' = \overline{\mathbf{0}}$. So provided a substring $\widetilde{\mathbf{s}} = \mathbf{0}$ (or $\widetilde{\mathbf{s}}_P = \mathbf{0}$), authentication does succeed with a probability 1/2. Via replay, one could increase the probability to 1. Eavesdropping on a single genuine protocol execution is required: $\mathbf{c}_S^{(1)}$, $\mathbf{c}_T^{(1)}$ and $\widetilde{\mathbf{s}}^{(1)}$. The malicious prover gets authenticated with the following: $\mathbf{c}_T^{(2)} \leftarrow \mathbf{c}_S^{(2)} \oplus \mathbf{c}_S^{(1)} \oplus \mathbf{c}_P^{(1)}$ and $\widetilde{\mathbf{s}}^{(2)} \leftarrow \widetilde{\mathbf{s}}^{(1)}$. One can easily replay old sessions keys as well. We stress that careful PRNG redesign can resolve all of the former.

5.14 Xu et al. (September 2012)

The protocol of Xu et al. [60] is represented by Figures 7(m) and 26. 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 $\mathbf{i}_0 = \mathbf{a} \oplus \mathbf{y}$.

[4] The proof-of-concept implementation employs 128-bit LFSRs, with $\mathbf{c}_S$ and $\mathbf{c}_T$ as the initial states, without specifying feedback polynomials. Furthermore, FPGA implementation results (Table III in [40] and Table 8 in [49]) strongly suggest the use of identical feedback polynomials.

	Token j	$\longleftrightarrow$	Server
$(1\times)$	R/W-secure MTP NVM: $\mathbf{i}_0 \leftarrow \mathbf{i}_{0j}$	$\longleftarrow$	$\mathbf{i}_{-1j}$, $\mathbf{i}_{0j} \leftarrow \mathit{TRNG}()$
	R/W-secure ROM: $\mathbf{y} \leftarrow \mathbf{y}'$	$\longleftarrow$	$\mathbf{y}' \leftarrow \mathit{TRNG}()$
	$\mathbf{r}_{1ij} \leftarrow \mathit{SPUF}(\mathbf{c}_{ij})$	$\longleftrightarrow$	$\langle \mathbf{c}_{ij}, \mathbf{r}_{1ij}, \mathbf{r}_{2ij}, \mathbf{r}_{3ij} \rangle$ with $\mathbf{c}_{ij} \leftarrow \mathit{TRNG}()$ and $i \in [1\ d]$
	$\mathbf{r}_{2ij} \leftarrow \mathit{SPUF}(\mathbf{r}_{1ij})$		
	$\mathbf{r}_{3ij} \leftarrow \mathit{SPUF}(\mathbf{r}_{2ij})$		
Auth. $(d\times)$		$\longleftarrow$	$\langle \mathbf{c}, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 \rangle \leftarrow \langle \mathbf{c}_{ij}, \mathbf{r}_{1ij}, \mathbf{r}_{2ij}, \mathbf{r}_{3ij} \rangle$ with $i \leftarrow d_j$
	$\mathbf{a} \leftarrow \mathbf{i}_0 \oplus \mathbf{y}$	$\xrightarrow{\mathbf{a}}$	
			If $\mathbf{i}_{0j} = \mathbf{a} \oplus \mathbf{y}'$
			$\quad \mathbf{b} \leftarrow \mathbf{i}_{0j} \oplus \mathbf{c}$
			Else if $\mathbf{i}_{-1j} = \mathbf{a} \oplus \mathbf{y}'$
			$\quad \mathbf{b} \leftarrow \mathbf{i}_{-1j} \oplus \mathbf{c}$
		$\xleftarrow{\mathbf{b}}$	Else Abort
	$\mathbf{c}' \leftarrow \mathbf{b} \oplus \mathbf{i}_0$		
	$\widetilde{\mathbf{r}}_1 \leftarrow \mathit{SPUF}(\mathbf{c}')$	$\xrightarrow{\widetilde{\mathbf{r}}_1}$	
			Abort if $\widetilde{\mathbf{r}}_1 \neq \mathbf{r}_1$
			If $\mathbf{i}_{0j} = \mathbf{a} \oplus \mathbf{y}'$
			$\quad \mathbf{i}_{-1j} \leftarrow \mathbf{i}_{0j}$
			$\quad \mathbf{i}_{0j} \leftarrow \mathbf{i}_{0j} \oplus \mathbf{r}_3$
		$\xleftarrow{\mathbf{r}_2}$	$d_j \leftarrow d_j - 1$
	$\widetilde{\mathbf{r}}_2 \leftarrow \mathit{SPUF}(\widetilde{\mathbf{r}}_1)$		
	Abort if $\widetilde{\mathbf{r}}_2 \neq \mathbf{r}_2$		
	$\widetilde{\mathbf{r}}_3 \leftarrow \mathit{SPUF}(\mathbf{r}_2)$		
	$\mathbf{i}_0 \leftarrow \mathbf{i}_0 \oplus \widetilde{\mathbf{r}}_3$		

Fig. 26. Authentication protocol of Xu et al. The desynchronization recovery logic is reconstructed to the best of our insights, given its incomplete description.

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 $\langle \mathbf{c}, \mathbf{r}_1 \rangle$.

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 $\mathbf{r}_2^{(2)} \leftarrow \mathbf{r}_2^{(1)}$.

5.15 He et al. (September 2012)

The protocol of He et al. [19] is represented by Figures 7(n) and 27. 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.

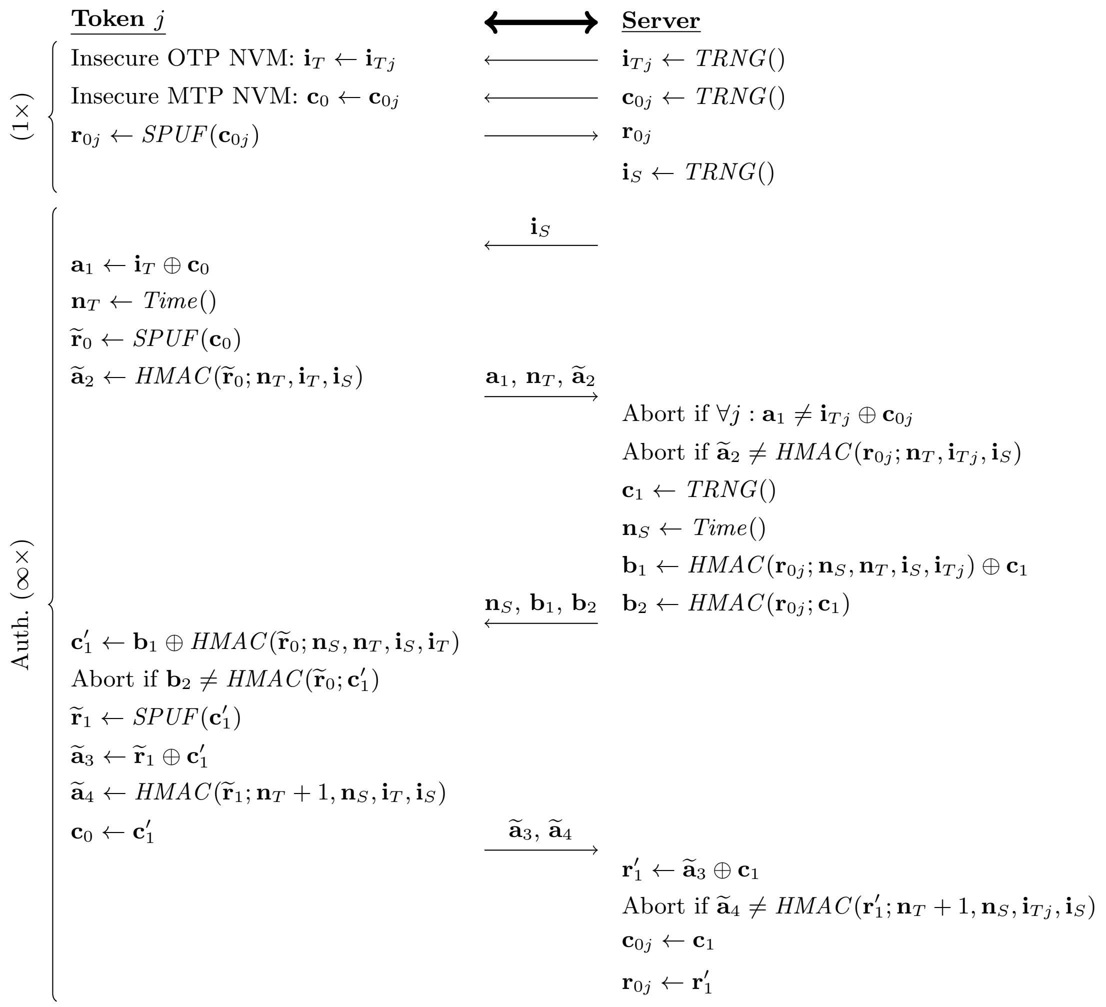
	Token j	$\longleftrightarrow$	**Server**
$(1\times)$	Insecure MTP NVM: $\mathbf{i} \leftarrow \mathbf{i}_j$	$\longleftarrow$	$\mathbf{i}_j \leftarrow \mathit{TRNG}()$
	Insecure MTP NVM: $\widetilde{\mathbf{x}}_0$		
	Insecure MTP NVM: $\mathbf{y} \leftarrow \mathbf{y}_j$	$\longleftarrow$	$\mathbf{y}_j \leftarrow \mathit{TRNG}()$
	$\mathbf{x}_{1j} \leftarrow \mathit{SPUF}(\widetilde{\mathbf{x}}_0)$	$\longrightarrow$	$\mathbf{x}_{1j}$
	$\mathbf{x}_{2j} \leftarrow \mathit{SPUF}(\mathbf{x}_{1j})$	$\longrightarrow$	$\mathbf{x}_{2j}$
Auth. $(\infty\times)$		$\longleftarrow$	
		$\xrightarrow{\mathbf{i}}$	
			Abort if $\forall j : \mathbf{i} \neq \mathbf{i}_j$
			$\mathbf{n}_1 \leftarrow \mathit{TRNG}()$
			$\mathbf{n}_2 \leftarrow \mathit{PRNG}(\mathbf{n}_1)$
			$\mathbf{b}_1 \leftarrow \mathbf{y}_j \oplus \mathbf{n}_1$
		$\xleftarrow{\mathbf{b}_1,\, \mathbf{b}_2}$	$\mathbf{b}_2 \leftarrow \mathbf{x}_{1j} \oplus \mathbf{n}_2$
	$\mathbf{n}'_1 \leftarrow \mathbf{b}_1 \oplus \mathbf{y}$		
	$\mathbf{n}'_2 \leftarrow \mathit{PRNG}(\mathbf{n}'_1)$		
	$\mathbf{x}'_1 \leftarrow \mathbf{b}_2 \oplus \mathbf{n}'_2$		
	Abort if $\mathbf{x}'_1 \neq \mathit{SPUF}(\widetilde{\mathbf{x}}_0)$		
	$\widetilde{\mathbf{x}}_2 \leftarrow \mathit{SPUF}(\mathbf{x}'_1)$		
	$\widetilde{\mathbf{x}}_3 \leftarrow \mathit{SPUF}(\widetilde{\mathbf{x}}_2)$		
	$\mathbf{n}'_3 \leftarrow \mathit{PRNG}(\mathbf{n}'_2)$		
	$\mathbf{n}'_4 \leftarrow \mathit{PRNG}(\mathbf{n}'_3)$		
	$\mathbf{n}'_5 \leftarrow \mathit{PRNG}(\mathbf{n}'_4)$		
	$\mathbf{n}'_6 \leftarrow \mathit{PRNG}(\mathbf{n}'_5)$		
	$\widetilde{\mathbf{a}}_1 \leftarrow \widetilde{\mathbf{x}}_2 \oplus \mathbf{n}'_3$		
	$\widetilde{\mathbf{a}}_2 \leftarrow \widetilde{\mathbf{x}}_3 \oplus \mathbf{n}'_4$		
	$\mathbf{i} \leftarrow \mathbf{i} \oplus \mathbf{n}'_5$		
	$\widetilde{\mathbf{x}}_0 \leftarrow \mathbf{x}'_1$		
	$\mathbf{y} \leftarrow \mathbf{y} \oplus \mathbf{n}'_6$	$\xrightarrow{\widetilde{\mathbf{a}}_1,\, \widetilde{\mathbf{a}}_2}$	
			$\mathbf{n}_3 \leftarrow \mathit{PRNG}(\mathbf{n}_2)$
			$\mathbf{x}'_2 \leftarrow \widetilde{\mathbf{a}}_1 \oplus \mathbf{n}_3$
			Abort if $\mathbf{x}'_2 \neq \mathbf{x}_{2j}$
			$\mathbf{n}_4 \leftarrow \mathit{PRNG}(\mathbf{n}_3)$
			$\mathbf{n}_5 \leftarrow \mathit{PRNG}(\mathbf{n}_4)$
			$\mathbf{n}_6 \leftarrow \mathit{PRNG}(\mathbf{n}_5)$
			$\mathbf{x}'_3 \leftarrow \widetilde{\mathbf{a}}_2 \oplus \mathbf{n}_4$
			$\mathbf{i}_j \leftarrow \mathbf{i}_j \oplus \mathbf{n}_5$
			$\mathbf{x}_{1j} \leftarrow \mathbf{x}_{2j}$
			$\mathbf{x}_{2j} \leftarrow \mathbf{x}'_3$
			$\mathbf{y}_j \leftarrow \mathbf{y}_j \oplus \mathbf{n}_6$

Fig. 27. Authentication protocol of He et al. Desynchronization recovery logic is not included due to its incomplete specification.

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 5.15.

5.16 Jung et al. (January 2013)

The protocol of Jung et al. [24] is represented by Figures 7(o) and 28. The authors make use of timestamps at both sides. The authors make use of the HMAC standard [32], 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 $\mathbf{a}_1$.



Token j
Server
$(1\times)$
Insecure OTP NVM: $\mathbf{i}_T \leftarrow \mathbf{i}_{Tj}$
$\mathbf{i}_{Tj} \leftarrow \mathit{TRNG}()$
Insecure MTP NVM: $\mathbf{c}_0 \leftarrow \mathbf{c}_{0j}$
$\mathbf{c}_{0j} \leftarrow \mathit{TRNG}()$
$\mathbf{r}_{0j} \leftarrow \mathit{SPUF}(\mathbf{c}_{0j})$
$\mathbf{r}_{0j}$
$\mathbf{i}_S \leftarrow \mathit{TRNG}()$
Auth. $(\infty\times)$
$\mathbf{i}_S$
$\mathbf{a}_1 \leftarrow \mathbf{i}_T \oplus \mathbf{c}_0$
$\mathbf{n}_T \leftarrow \mathit{Time}()$
$\widetilde{\mathbf{r}}_0 \leftarrow \mathit{SPUF}(\mathbf{c}_0)$
$\widetilde{\mathbf{a}}_2 \leftarrow \mathit{HMAC}(\widetilde{\mathbf{r}}_0; \mathbf{n}_T, \mathbf{i}_T, \mathbf{i}_S)$
$\mathbf{a}_1$, $\mathbf{n}_T$, $\widetilde{\mathbf{a}}_2$
Abort if $\forall j : \mathbf{a}_1 \neq \mathbf{i}_{Tj} \oplus \mathbf{c}_{0j}$
Abort if $\widetilde{\mathbf{a}}_2 \neq \mathit{HMAC}(\mathbf{r}_{0j}; \mathbf{n}_T, \mathbf{i}_{Tj}, \mathbf{i}_S)$
$\mathbf{c}_1 \leftarrow \mathit{TRNG}()$
$\mathbf{n}_S \leftarrow \mathit{Time}()$
$\mathbf{b}_1 \leftarrow \mathit{HMAC}(\mathbf{r}_{0j}; \mathbf{n}_S, \mathbf{n}_T, \mathbf{i}_S, \mathbf{i}_{Tj}) \oplus \mathbf{c}_1$
$\mathbf{b}_2 \leftarrow \mathit{HMAC}(\mathbf{r}_{0j}; \mathbf{c}_1)$
$\mathbf{n}_S$, $\mathbf{b}_1$, $\mathbf{b}_2$
$\mathbf{c}'_1 \leftarrow \mathbf{b}_1 \oplus \mathit{HMAC}(\widetilde{\mathbf{r}}_0; \mathbf{n}_S, \mathbf{n}_T, \mathbf{i}_S, \mathbf{i}_T)$
Abort if $\mathbf{b}_2 \neq \mathit{HMAC}(\widetilde{\mathbf{r}}_0; \mathbf{c}'_1)$
$\widetilde{\mathbf{r}}_1 \leftarrow \mathit{SPUF}(\mathbf{c}'_1)$
$\widetilde{\mathbf{a}}_3 \leftarrow \widetilde{\mathbf{r}}_1 \oplus \mathbf{c}'_1$
$\widetilde{\mathbf{a}}_4 \leftarrow \mathit{HMAC}(\widetilde{\mathbf{r}}_1; \mathbf{n}_T + 1, \mathbf{n}_S, \mathbf{i}_T, \mathbf{i}_S)$
$\mathbf{c}_0 \leftarrow \mathbf{c}'_1$
$\widetilde{\mathbf{a}}_3$, $\widetilde{\mathbf{a}}_4$
$\mathbf{r}'_1 \leftarrow \widetilde{\mathbf{a}}_3 \oplus \mathbf{c}_1$
Abort if $\widetilde{\mathbf{a}}_4 \neq \mathit{HMAC}(\mathbf{r}'_1; \mathbf{n}_T + 1, \mathbf{n}_S, \mathbf{i}_{Tj}, \mathbf{i}_S)$
$\mathbf{c}_{0j} \leftarrow \mathbf{c}_1$
$\mathbf{r}_{0j} \leftarrow \mathbf{r}'_1$

Fig. 28. Authentication protocol of Jung et al.

Incomplete Specification (#1) An implementation of $\mathit{Time}()$ 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.

5.17 Lee et al. II (July 2013)

The second protocol of Lee et al. [36] is represented by Figures 7(p) and 29. Exposure of the token state ($\mathbf{i}$, $\mathbf{c}_0$ and $\mathbf{y}$) via physical leakage is claimed not to be a threat.

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 a 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.

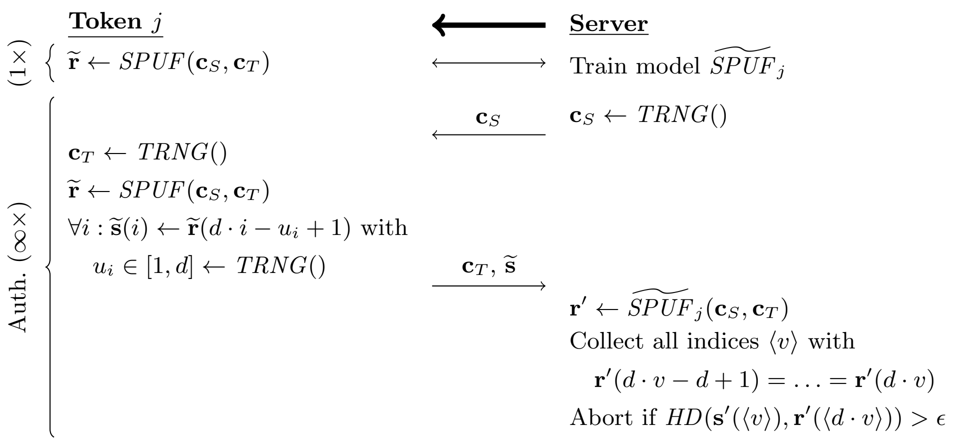
Token/Server Impersonation and Denial-of-Service via Leakage (#8) We argue that leakage leads to full system disclosure. Assume an attacker to observe a genuine protocol run (1). Via state leakage, one subsequently obtains $\mathbf{i}$, $\mathbf{c}_0^{(2)}$ and $\mathbf{y}$. First consider impersonation 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 $\widetilde{\mathbf{a}}_1$.

	Token j	$\longleftrightarrow$	**Server**
$(1\times)$	Insecure OTP NVM: $\mathbf{i} \leftarrow \mathbf{i}_j$	$\longleftarrow$	$\mathbf{i}_j \leftarrow \mathit{TRNG}()$
	Insecure MTP NVM: $\mathbf{c}_0 \leftarrow \mathbf{c}_{0j}$	$\longleftarrow$	$\mathbf{c}_{0j} \leftarrow \mathit{TRNG}()$
	Insecure OTP MVM: $\mathbf{y} \leftarrow \mathbf{y}_j$	$\longleftarrow$	$\mathbf{y}_j \leftarrow \mathit{TRNG}()$
	$\mathbf{r}_{0j} \leftarrow \mathit{SPUF}(\mathbf{c}_{0j})$	$\longrightarrow$	$\mathbf{x}_{0j} \leftarrow \mathit{Hash}(\mathbf{r}_{0j})$
Auth. $(\infty\times)$			$\mathbf{n} \leftarrow \mathit{TRNG}()$
		$\xleftarrow{\mathbf{b}_1}$	$\mathbf{b}_1 \leftarrow \mathbf{i}_j \oplus \mathbf{n}$
	$\widetilde{\mathbf{r}}_0 \leftarrow \mathit{SPUF}(\mathbf{c}_0)$		
	$\mathbf{c}_1 \leftarrow \mathit{Hash}(\mathbf{c}_0)$		
	$\widetilde{\mathbf{r}}_1 \leftarrow \mathit{SPUF}(\mathbf{c}_1)$		
	$\mathbf{n}' \leftarrow \mathbf{b}_1 \oplus \mathbf{i}$		
	$\widetilde{\mathbf{a}}_1 \leftarrow \widetilde{\mathbf{r}}_0 \oplus \mathbf{c}_0$		
	$\widetilde{\mathbf{a}}_2 \leftarrow \mathit{Hash}(\widetilde{\mathbf{r}}_1) \oplus \mathbf{c}_0$		
	$\widetilde{\mathbf{a}}_3 \leftarrow \mathit{MAC}(\mathbf{c}_0; \widetilde{\mathbf{a}}_1, \widetilde{\mathbf{a}}_2, \mathbf{n}')$	$\xrightarrow{\widetilde{\mathbf{a}}_1,\, \widetilde{\mathbf{a}}_2,\, \widetilde{\mathbf{a}}_3}$	
			Abort if $\widetilde{\mathbf{a}}_3 \neq \mathit{MAC}(\mathbf{c}_{0j}; \widetilde{\mathbf{a}}_1, \widetilde{\mathbf{a}}_2, \mathbf{n})$
			$\mathbf{r}'_0 \leftarrow \widetilde{\mathbf{a}}_1 \oplus \mathbf{c}_{0j}$
			Abort if $\mathbf{x}_{0j} \neq \mathit{Hash}(\mathbf{r}'_0)$
			$\mathbf{b}_2 \leftarrow \mathbf{y}_j \oplus \mathbf{c}_{0j}$
			$\mathbf{b}_3 \leftarrow \mathit{MAC}(\mathbf{c}_{0j}; \mathbf{b}_2, \mathbf{n})$
			$\mathbf{x}_{0j} \leftarrow \widetilde{\mathbf{a}}_2 \oplus \mathbf{c}_{0j}$
		$\xleftarrow{\mathbf{b}_2,\, \mathbf{b}_3}$	$\mathbf{c}_{0j} \leftarrow \mathit{Hash}(\mathbf{c}_{0j})$
	Abort if $\mathbf{b}_3 \neq \mathit{MAC}(\mathbf{c}_0; \mathbf{b}_2, \mathbf{n}')$		
	Abort if $\mathbf{y} \neq \mathbf{b}_2 \oplus \mathbf{c}_0$		
	$\mathbf{c}_0 \leftarrow \mathbf{c}_1$		

Fig. 29. 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.

5.18 Noise Bifurcation (May 2014)

The noise bifurcation proposal [61] can be understood as a minor variation on the slender PUF protocol, as also reflected by Figure 7(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 30 represents the protocol.



Token j
Server
$(1\times)$
$\widetilde{\mathbf{r}} \leftarrow SPUF(\mathbf{c}_S, \mathbf{c}_T)$
Train model $\widetilde{SPUF}_j$
$\mathbf{c}_S$
$\mathbf{c}_S \leftarrow TRNG()$
Auth. $(\infty\times)$
$\mathbf{c}_T \leftarrow TRNG()$
$\widetilde{\mathbf{r}} \leftarrow SPUF(\mathbf{c}_S, \mathbf{c}_T)$
$\forall i : \widetilde{\mathbf{s}}(i) \leftarrow \widetilde{\mathbf{r}}(d \cdot i - u_i + 1)$ with
$u_i \in [1, d] \leftarrow TRNG()$
$\mathbf{c}_T, \widetilde{\mathbf{s}}$
$\mathbf{r}' \leftarrow \widetilde{SPUF}_j(\mathbf{c}_S, \mathbf{c}_T)$
Collect all indices $\langle v \rangle$ with
$\mathbf{r}'(d \cdot v - d + 1) = \ldots = \mathbf{r}'(d \cdot v)$
Abort if $HD(\mathbf{s}'(\langle v \rangle), \mathbf{r}'(\langle d \cdot v \rangle)) > \epsilon$

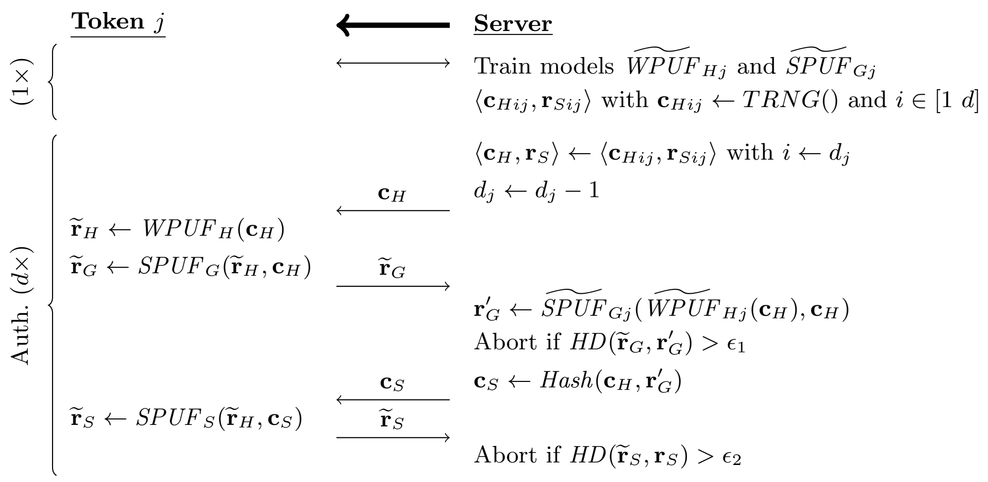
Fig. 30. Noise bifurcation protocol.

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) Instantiating the protocol is difficult in practice, compared to other proposals. 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 and retrieve indices u_i (and/or v) as such. For a monolithic PUF, this would be an extremely difficult balancing exercise. For composite PUFs, such as the proposed XOR construction, it works out though. During enrollment, the arbiter chains can be modeled separately, bypassing the XOR via one-time interfaces. Specialized attacks might estimate a lower bound for the number of chains that needs to be XORed. In the article, the authors showed how to defeat the protocol using simulated arbiter XOR PUFs with up to four chains. It is difficult to obtain a safe estimate though, as one can never exclude the possibility that an alternative attack might perform better. In addition, XORing amplifies noisiness, posing a practical upper bound on the number of chains.

5.19 System of PUFs (October 2014)

The system of PUFs proposal [31] consists of three PUFs, as shown in Figure 7(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 31 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.



Token j
Server
$(1\times)$
Train models $\widetilde{WPUF}_{Hj}$ and $\widetilde{SPUF}_{Gj}$
$\langle \mathbf{c}_{Hij}, \mathbf{r}_{Sij} \rangle$ with $\mathbf{c}_{Hij} \leftarrow TRNG()$ and $i \in [1\ d]$
Auth. $(d\times)$
$\langle \mathbf{c}_H, \mathbf{r}_S \rangle \leftarrow \langle \mathbf{c}_{Hij}, \mathbf{r}_{Sij} \rangle$ with $i \leftarrow d_j$
$\mathbf{c}_H$
$d_j \leftarrow d_j - 1$
$\widetilde{\mathbf{r}}_H \leftarrow WPUF_H(\mathbf{c}_H)$
$\widetilde{\mathbf{r}}_G \leftarrow SPUF_G(\widetilde{\mathbf{r}}_H, \mathbf{c}_H)$
$\widetilde{\mathbf{r}}_G$
$\mathbf{r}'_G \leftarrow \widetilde{SPUF}_{Gj}(\widetilde{WPUF}_{Hj}(\mathbf{c}_H), \mathbf{c}_H)$
Abort if $HD(\widetilde{\mathbf{r}}_G, \mathbf{r}'_G) > \epsilon_1$
$\mathbf{c}_S$
$\mathbf{c}_S \leftarrow Hash(\mathbf{c}_H, \mathbf{r}'_G)$
$\widetilde{\mathbf{r}}_S \leftarrow SPUF_S(\widetilde{\mathbf{r}}_H, \mathbf{c}_S)$
$\widetilde{\mathbf{r}}_S$
Abort if $HD(\widetilde{\mathbf{r}}_S, \mathbf{r}_S) > \epsilon_2$

Fig. 31. 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 [56], 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. In Figure 8, 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 original strong PUF versions both claimed to offer a considerable side-channel advantage. However, the same physical attacks still apply and one only needs to introduce an additional machine learning step. We do not exclude the possibility that other protocols might leverage the strong PUF side-channel benefit more effectively and suggest this analysis as further work.

The second category comprehends **strong PUF obfuscation**. Although not equally secure against modeling attacks, 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 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. SBO 121552.

		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× 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'.

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)*, pages 495–499, 2011.
2. G. T. Becker. On the pitfalls of using arbiter pufs as building blocks. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 34(8):1295–1307, 2015.
3. G. T. Becker and R. Kumar. Active and passive side-channel attacks on delay based PUF designs. *IACR Cryptology ePrint Archive*, 2014:287, 2014.
4. L. Bolotnyy and G. Robins. Physically unclonable function-based security and privacy in RFID systems. In *5th International Conference on Pervasive Computing and Communications (PerCom 2007)*, pages 211–220, 2007.
5. X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure remote authentication using biometric data. In *Advances in Cryptology - EUROCRYPT 2005, 24th International Conference on the Theory and Applications of Cryptographic Techniques*, pages 147–163, 2005.
6. P. F. Cortese, F. Gemmiti, B. Palazzi, M. Pizzonia, and M. Rimondini. Efficient and practical authentication of puf-based rfid tags in supply chains. In *International Conference on RFID-Technology and Applications (RFID-TA 2010)*, pages 182–188, June 2010.
7. A. Das, Ü. Kocabaş, A. Sadeghi, and I. Verbauwhede. PUF-based secure test wrapper design for cryptographic SoC testing. In *Conference & Exhibition on Design, Automation & Test in Europe (DATE 2012)*, pages 866–869, 2012.
8. J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede. Helper data algorithms for puf-based key generation: Overview and analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(6):889–902, 2015.
9. J. Delvaux and I. Verbauwhede. Attacking puf-based pattern matching key generators via helper data manipulation. In *Topics in Cryptology - CT-RSA 2014, The Cryptographers' Track at the RSA Conference 2014*, pages 106–131, 2014.
10. S. Devadas, G. 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. In *International Conference on RFID*, pages 58–64, 2008.
11. 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.
12. 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*, pages 523–540, 2004.
13. B. Gassend. Physical random functions. Master's thesis, Massachusetts Institute of Technology, 2003.
14. B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas. Controlled physical random functions. In *ACSAC*, pages 149–160. IEEE Computer Society, 2002.
15. 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.
16. B. Gassend, M. van Dijk, D. E. Clarke, E. Torlak, S. Devadas, and P. Tuyls. Controlled physical random functions and applications. *ACM Trans. Inf. Syst. Secur.*, 10(4), 2008.
17. J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls. FPGA intrinsic PUFs and their use for IP protection. In *9th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2007)*, pages 63–80, 2007.
18. G. Hammouri, E. Öztürk, and B. Sunar. A tamper-proof and lightweight authentication scheme. *Pervasive and Mobile Computing*, 4(6):807–818, 2008.
19. Z. He and L. Zou. High-efficient RFID authentication protocol based on physical unclonable function. In *8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2012)*, pages 1–4, Sept 2012.
20. C. Helfmeier, C. Boit, D. Nedospasov, and J. Seifert. Cloning physically unclonable functions. In *International Symposium on Hardware-Oriented Security and Trust (HOST 2013)*, pages 1–6, 2013.
21. J. Hermans, R. Peeters, and J. Fan. IBIHOP: Proper Privacy Preserving Mutual RFID Authentication. In *Workshop on RFID and IoT Security - RFIDSec Asia 2013*, pages 45–56, 2013.
22. 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.
23. Y. Jin, W. Xin, H. Sun, and Z. Chen. PUF-based RFID authentication protocol against secret key leakage. In *14th Asia-Pacific Web Conference on Web Technologies and Applications (APWeb 2012)*, pages 318–329, 2012.
24. S. W. Jung and S. Jung. HRP: A HMAC-based RFID mutual authentication protocol using PUF. In *International Conference on Information Networking (ICOIN 2013)*, pages 578–582, Jan 2013.

25. D. Karakoyunlu and B. Sunar. Differential template attacks on puf enabled cryptographic devices. In *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, pages 1–6, Dec 2010.
26. 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.
27. 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.
28. S. Katzenbeisser, Ü. Kocabaş, V. van der Leest, A. Sadeghi, G.-J. Schrijen, and C. Wachsmann. Recyclable pufs: logically reconfigurable pufs. *J. Cryptographic Engineering*, 1(3):177–186, 2011.
29. S. Katzenbeisser, Ünal Kocabaş, V. van der Leest, A. Sadeghi, G.-J. Schrijen, H. Schröder, and C. Wachsmann. Recyclable PUFs: Logically reconfigurable PUFs. In *13th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2011)*, pages 374–389, 2011.
30. Ünal. Kocabaş, A. Peter, S. Katzenbeisser, and A. Sadeghi. Converse puf-based authentication. In *5th International Conference on Trust and Trustworthy Computing (TRUST 2012)*, pages 142–158, 2012.
31. S. T. C. Konigsmark, L. K. Hwang, D. Chen, and M. D. F. Wong. System-of-pufs: Multilevel security for embedded systems. In *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS 2014)*, pages 1–10, 2014.
32. H. Krawczyk, M. Bellare, and R. Canetti. Hmac: Keyed-hashing for message authentication. Technical report, 1997.
33. L. Kulseng, Z. Yu, Y. Wei, and Y. Guan. Lightweight mutual authentication and ownership transfer for RFID systems. In *29th International Conference on Computer Communications (INFOCOM 2010)*, pages 251–255, 2010.
34. 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.
35. Y. S. Lee, T. Y. Kim, and H. J. Lee. Mutual authentication protocol for enhanced rfid security and anti-counterfeiting. In *26th International Conference on Advanced Information Networking and Applications Workshops (WAINA 2012)*, pages 558–563, March 2012.
36. Y. S. Lee, H. J. Lee, and E. Alasaarela. Mutual authentication in wireless body sensor networks (WBSN) based on physical unclonable function (PUF). In *9th International Wireless Communications and Mobile Computing Conference (IWCMC 2013)*, pages 1314–1318, July 2013.
37. R. Maes. *Physically Unclonable Functions: Constructions, Properies and Applications*. PhD thesis, KU Leuven, 2012.
38. R. Maes. An accurate probabilistic reliability model for silicon pufs. In *15th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2013)*, pages 73–89, 2013.
39. M. Majzoobi, F. Koushanfar, and M. Potkonjak. Testing techniques for hardware security. In *International Test Conference (ITC 2008)*, pages 1–10, 2008.
40. 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 *Symposium on Security and Privacy Workshops*, pages 33–44, 2012.
41. D. Merli, D. Schuster, F. Stumpf, and G. Sigl. Semi-invasive em attack on fpga ro pufs and countermeasures. In *6th Workshop on Embedded Systems Security (WESS 2011)*, 2011.
42. D. Merli, D. Schuster, F. Stumpf, and G. Sigl. Side-channel analysis of PUFs and fuzzy extractors. In *4th International Conference on Trust and Trustworthy Computing (TRUST 2011)*, pages 33–47, 2011.
43. M. A. Orumiehchiha, J. Pieprzyk, and R. Steinfeld. Breaking nlm-mac generator. Cryptology ePrint Archive, Report 2013/202, 2013. http://eprint.iacr.org/.
44. E. Öztürk, G. Hammouri, and B. Sunar. Towards robust low cost authentication for pervasive devices. In *PerCom*, pages 170–178. IEEE Computer Society, 2008.
45. R. S. Pappu. *Physical One-Way Functions*. PhD thesis, MIT, 2001.
46. 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.
47. D. C. Ranasinghe, D. W. Engels, and P. H. Cole. Security and privacy: Modest proposals for low-cost rfid systems. In *Auto-ID Labs Research Workshop*, 2004.
48. M. M. Riedel. Random permutation statistics, 2014.
49. M. Rostami, M. Majzoobi, F. Koushanfar, D. S. Wallach, and S. Devadas. Robust and reverse-engineering resilient PUF authentication and key-exchange by substring matching. *IEEE Trans. Emerging Topics Comput.*, 2(1):37–49, 2014.
50. U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber. Modeling attacks on physical unclonable functions. In *17th Conference on Computer and Communications Security (CCS 2010)*, pages 237–249, 2010.

51. 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.
52. U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, M. Majzoobi, F. Koushanfar, and W. P. Burleson. Efficient power and timing side channels for physical unclonable functions. In *16th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2014)*, pages 476–492, 2014.
53. A. Sadeghi, I. Visconti, and C. Wachsmann. Enhancing RFID security and privacy by physically unclonable functions. In *Towards Hardware-Intrinsic Security - Foundations and Practice*, pages 281–305. 2010.
54. M. Safkhani, N. Bagheri, and M. Naderi. Security analysis of a PUF based RFID authentication protocol. *IACR Cryptology ePrint Archive*, 2011:704, 2011.
55. 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.
56. G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *44th Design Automation Conference (DAC 2007)*, pages 9–14, 2007.
57. S. Tajik, E. Dietz, S. Frohmann, J. Seifert, D. Nedospasov, C. Helfmeier, C. Boit, and H. Dittrich. Physical characterization of arbiter pufs. In *16th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2014)*, pages 493–509, 2014.
58. 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*, pages 115–131, 2006.
59. A. Van Herrewege, S. Katzenbeisser, R. Maes, R. Peeters, A. Sadeghi, I. Verbauwhede, and C. Wachsmann. Reverse fuzzy extractors: Enabling lightweight mutual authentication for puf-enabled rfids. In *16th International Conference on Financial Cryptography and Data Security (FC 2012)*, pages 374–389, 2012.
60. Y. Xu and Z. He. Design of a security protocol for low-cost rfid. In *8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2012)*, pages 1–3, 2012.
61. M. M. Yu, D. M'Raïhi, I. Verbauwhede, and S. Devadas. A noise bifurcation architecture for linear additive physical functions. In *International Symposium on Hardware-Oriented Security and Trust (HOST 2014)*, pages 124–129, 2014.