A Lockdown Technique to Prevent Machine Learning on PUFs for Lightweight Authentication

Meng-Day (Mandel) Yu[1,3,5], Matthias Hiller[2], Jeroen Delvaux[3,4], Richard Sowell[1],
Srinivas Devadas[5], Ingrid Verbauwhede[3]
[1]Verayo, Inc., USA {*myu,rsowell*}*@verayo.com*
[2]Institute for Security in Information Technology at Technische Universität München, Germany *matthias.hiller@tum.de*
[3]Computer Security & Industrial Cryptography (COSIC) lab at KU Leuven and iMinds, Belgium {*jeroen.delvaux,ingrid.verbauwhede*}*@esat.kuleuven.be*
[4]Information Security Lab at Shanghai Jiao Tong University, China
[5]Computer Science & Artificial Intelligence Lab (CSAIL) at MIT, USA *devadas@mit.edu*

Abstract. We present a lightweight PUF-based authentication approach that is practical in settings where a server authenticates a device, and for use cases where the number of authentications is limited over a device's lifetime. Our scheme uses a server-managed challenge/response pair (CRP) *lockdown* protocol: unlike prior approaches, an adaptive chosen-challenge adversary with machine learning capabilities *cannot* obtain new CRPs without the server's implicit permission. The adversary is faced with the problem of deriving a PUF model with a limited amount of machine learning training data. Our system-level approach allows a so-called strong PUF to be used for lightweight authentication in a manner that is *heuristically secure* against today's best machine learning methods through a worst-case CRP exposure algorithmic validation. We also present a degenerate instantiation using a weak PUF that is secure against *computationally unrestricted* adversaries, which includes *any* learning adversary, for practical device lifetimes and read-out rates. We validate our approach using silicon PUF data, and demonstrate the feasibility of supporting 10, 1000, and 1M authentications, including practical configurations that are not learnable with polynomial resources, e.g., the number of CRPs and the attack runtime, using recent results based on the *probably-approximately-correct* (PAC) complexity-theoretic framework.

Keywords: Physical Unclonable Function, Authentication, Machine Learning, Heuristic Security, Computationally Unrestricted Adversary, Probably Approximately Correct (PAC) Learning

1 Introduction

We consider a common authentication scenario between a low-cost resource-constrained device and resource-rich server, in which secret keys typically form the backbone and must be stored securely. Unfortunately, *non-volatile memory* (NVM) tends to be vulnerable to physical attacks. Cast in silicon by MIT in the early 2000s [1], [2], [3], *physically unclonable functions* (PUFs) were envisioned to offer improved physical security by not requiring static (always present) secret keys on the device. Silicon PUFs harvest manufacturing variability to produce device-unique but noisy bits. These bits, which are generated dynamically (present only when needed) may be error-corrected and hashed into a secret key that is subsequently used with a cryptographic algorithm such as a block cipher or a keyed hash function for authentication purposes.

Nevertheless, PUF-based key generation may be too *heavyweight* for some use cases. In a recent survey of nineteen PUF-based entity authentication protocols [4], only two finalists [5], [6] require neither an error-correction code nor a cryptographic algorithm. However, these rely on a true random number generator (TRNG) instead and offer a form of *heuristic security* against so-called modeling attacks that is difficult to validate – the adversary's capability in terms of available challenge/response pair (CRP) machine learning training material is *difficult to upper-bound*. Also the non-surveyed super-high information-content (SHIC) protocol [7], although secure against *computationally unrestricted* adversaries, has practical limitations (cf. Sec. 2.3).

Indeed, without a cryptographic algorithm and a secret or private key, it is difficult to derive an exponential number of CRPs from a linearly-sized circuit. Early MIT PUF researchers [1] envisioned lightweight authentication to be performed using a threshold-based comparison where *no error correction* – and by extension *no cryptographic algorithm* – is required. Unfortunately, arbitrary logical or arithmetic post-processing cannot be applied to the silicon manufacturing variation since the physical noise would be amplified. As a result, popular PUF authentication circuits are evaluated in a mostly linear fashion, with limited non-linear mixing, and are therefore prone to modeling attacks. A decade after the initial MIT results, in his PhD thesis, Maes regarded a practical instantiation of the envisioned PUF authentication circuit as an open problem [8].

1.1 Contribution

We continue the pursuit for true lightweight entity authentication. We observe that prior approaches have an "open

M. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, I. Verbauwhede, "A Lockdown Technique to Prevent Machine Learning on PUFs for Lightweight Authentication," in IEEE Transactions on Multi-Scale Computing Systems, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7450665. DOI: 10.1109/TMSCS.2016.2553027.

interface" that can be freely queried by the adversary to obtain new CRPs. We do not attempt to create a new challenge/response PUF circuit and argue security in the face of arbitrary CRP exposure, which is emblematic of prior approaches [5], [6], [9], [10]. Instead, we take a practical *protocol-level approach to limit the availability CRP material* that an adaptive chosen-challenge machine-learning-equipped adversary may obtain. The trade-off is that we can support only a limited number of authentications for the verifier, e.g., 10, 1000, or 10^6, pre-specified depending on the use case. We emphasize that the adversary can still make arbitrary adaptive chosen-challenge queries against the device, but no new CRPs are obtained by the adversary unless implicitly permitted by the verifier/server.

Our basic *lockdown* protocol requires neither error-correction nor a cryptographic algorithm nor a TRNG on the device. Even without TRNG, we are still heuristically secure against today's best modeling attacks based on a direct heuristic validation in terms of the *worst-case CRP exposure*; without lockdown, it is difficult to upper-bound the adversary's capability in terms of available CRP material. Also, we present a PRNG design of which the security can be guaranteed in advance via an exhaustive check. Furthermore, our approach is compatible with almost every PUF. We eliminate SHIC limitations, while maintaining security against computationally unrestricted adversaries.

Our extended protocol, despite introducing a TRNG, offers three benefits: i) The authentication is mutual rather than unilateral; ii) We improve resistance against recent machine learning attacks exploiting noise side-channel information; iii) Our protocol may inherit the benefits of prior protocols such as slender PUF [5], i.e., obfuscation techniques to further improve the modeling robustness.

We also present *practically realizable* PUF instantiations that cannot be learned with resources polynomial with respect to the circuit size based on recent PUF learning results [11] using *probably-approximately-correct* (PAC) theory [12]. Previously, non-PAC-learnable instantiations were thought to be infeasible. Our extended protocol (cf. Sec. 4) plus our improved noise equation (cf. Eqn. 6) and response lengthening analysis (cf. Sec. 9.3) that were missing in [11] make this result possible, and we include silicon validation (cf. Sec 7.1, Sec. 7.2) to demonstrate practical feasibility.

Our main contributions are summarized below.

- We propose the first strong PUF authentication scheme where the adversary's power is *upper-bounded* in terms of available CRPs (cf. Sec. 3.2.1).
- We propose the first weak PUF authentication scheme using SRAM that is secure against *computationally unrestricted* adversaries (cf. Sec. 8.2).
- We are the first to confirm the existence of an exponentially hard-to-learn PUF system-level instantiation *that is practically realizable*, with the hardness based on PAC learning theory [11] (cf. Sec. 9.3).

1.2 Outline

Section 2 contains preliminaries. Section 3 introduces the lockdown protocol and Section 4 describes an extension to prevent attacks requiring repeated measurements. Section 5 elaborates the PRNG design. Section 6 describes protocol instantiation values. Section 7 validates our approach with silicon data and presents an analysis of machine learning attack results. Section 8 presents architecture examples. Section 9 makes comparisons against prior proposals and theoretical learning results. We conclude the work in Section 10.

2 Preliminaries

2.1 Notation

Binary vectors are denoted with a bold character, e.g., $\mathbf{r}$. Concatenation and bit-wise XORing are denoted with $\|$ and $\oplus$ respectively. Functions are printed in a sans-serif font, e.g., Hamming weight $\mathsf{HW}(\mathbf{r})$ counts the number of 1's in vector $\mathbf{r}$. Let $\mathsf{L}(\mathbf{r})$ denote the length of $\mathbf{r}$, as used in, e.g., the fractional Hamming distance $\mathsf{FHD}(\widetilde{\mathbf{r}}, \mathbf{r}) = \mathsf{HW}(\widetilde{\mathbf{r}} \oplus \mathbf{r})/\mathsf{L}(\mathbf{r})$.

We use angle brackets $\langle . \rangle$ to denote an indexed list. We use round brackets to denote a tuple, e.g., $(\mathbf{c}, \mathbf{r})$ refers to a challenge $\mathbf{c}$ and the response bits $\mathbf{r}$ that are derived from it. Let $\mathsf{L}(\langle \mathbf{c} \rangle)$ denote the number of sub-challenges derived from the starting challenge $\mathbf{c}$.

2.2 Strong and Weak PUFs

Silicon PUFs can be divided into two categories: *strong PUFs* and *weak PUFs*. Both derive their functional behavior from a limited number of circuit components, all prone to *silicon manufacturing variation*. A weak PUF derives a small number of CRPs, scaling linearly with circuit size.

The goal of an *ideal* strong PUF [13], [14] is lofty. It attempts to produce an exponential number of CRPs from a linear number of circuit components where the response bits are difficult to predict. Practical strong PUFs are composed of linear circuits in order to prevent physical noise amplification and are therefore prone to modeling attacks, e.g., using machine learning [14], [15], [16], [17], [18].

While a practical instantiation of a strong PUF circuit has been declared as an open problem [8], an *ideal* weak PUF might be approximated well in practice. E.g., using an SRAM PUF [13], where each PUF bit is derived from a physically distinct component thereby offering an opportunity for any bit correlations to be minimized via careful layout.

2.3 Super High Information Content (SHIC)

A Super High Information Content (SHIC) PUF consists of an *extreme large* memory (e.g., $> 10^{10}$ bits) with *extremely slow* read-out (e.g., 100 bits/second). The corresponding CRP protocol is secure against *computationally unrestricted* adversaries [7]. The SHIC system takes a "brute-force" approach to behave like an ideal strong PUF: it is not feasible for an adversary to read a sufficiently large subset of the memory locations (response bits) in a practical amount of time. This is despite adversarial computational power. The authors of [7] argue that their system can be implemented using emerging high-density crossbar memory, which does not integrate well with conventional CMOS designs.

2.4 Machine Learning and Side-Channel Attacks

Machine learning attacks on strong PUFs were proposed in [19], [20] and certain constructs broken in [14], [17], [18], [21], [22], [23]. Recent machine learning attacks exploited

PUF noise measurements as side-channel information [18], [24], [25] or performed noise filtering to improve the signal-to-noise ratio [23], [26], [27]. These recently published results use *repeated measurements* to obtain PUF noise information.

Our extended protocol provides resistance against attacks that require repeated measurements *in general*. This also includes a recent *backside photonic* attack [28] that requires millions of repeated measurements. This attack requires challenges with low pair-wise Hamming distances which may not be present in practice. The recent lattice basis reduction attack [29] removes the low Hamming distance constraint but still requires backside photonic access.

We shall demonstrate how we can take an XOR PUF circuit that is learnable, and *still* instantiate it at a system level and use it in a manner that is heuristically secure against the best published machine learning attacks to date (cf. Sec. 7.3).

2.5 Machine Learning Complexity Theory Results

In Valiant's seminal work [12], a framework was created to relate machine learning to complexity theory. Recent theoretical machine learning results from [11] based on Valient's PAC theory concluded certain XOR PUF constructions to be learnable in polynomial time given a polynomial number of CRPs; other constructions were declared non-PAC-learnable and it was also suggested that these were also *infeasible in practice* due to PUF noise.

A natural question arises as to whether there exists a PUF instantiation that is exponentially difficult to learn according to [11] and yet is *practically realizable*. We are the first to provide an answer in the affirmative.

2.6 Adversary Model

The enrollment occurs in a secure environment. Afterwards, the adversary has access to the device's interface and is free to brute-force query it, with challenges possibly adaptively chosen. The obtained CRP information is used for algorithmic machine learning attacks. Eavesdropping, manipulation and replay of protocol traffic are all deemed possible. The server has secure storage available and its computations are shielded from the outside world. Despite introducing a general-purpose countermeasure against repeated measurements, side-channel attacks that rely on more specific implementation aspects [30] are outside our scope of work.

2.7 Basic Strong PUF Authentication Protocol

For the basic strong PUF authentication protocol [9], each device embeds a strong PUF (SPUF) that is used for d unilateral authentication events. In Fig. 1, we look at device i (prover) and the server (verifier). A challenge $\mathbf{c}$ is applied, and a list of sub-challenges $\langle \mathbf{c} \rangle$ is derived on the device using a PRNG, where each sub-challenge is used to evaluate the device-unique manufacturing variation associated with SPUF to generate a single response bit; these single bit values are concatenated together to form $\widetilde{\mathbf{r}}$. During an initial one-time enrollment, the server collects a database comprising of d securely stored $(\mathbf{c}_{ij}, \mathbf{r}_{ij})$ entries associated with d authentication events. A device identifier $\mathbf{id}_i$, e.g., a serial number, is stored on chip in an one-time programmable storage (OTP-S) that is publicly readable and does not need to be kept secret; e-fuse technology can be used.

Device i		**Server**
Enrollment ($1\times$)		
OTP-S: $\mathbf{id} \leftarrow \mathbf{id}_i$	$\xleftarrow{\mathbf{id}_i}$	$\mathbf{id}_i \leftarrow \mathsf{SerialNumber}()$
	$\longleftrightarrow$	$(\mathbf{c}_{ij}, \mathbf{r}_{ij})$ with $j \in [1, d]$ and $\mathbf{c}_{ij} \leftarrow \mathsf{TRNG}()$
		$j_i \leftarrow 0$
Authentication ($d\times$)		
	$\xleftarrow{\mathsf{init}}$	
	$\xrightarrow{\mathbf{id}}$	Abort if $\forall i : \mathbf{id} \neq \mathbf{id}_i$
		$j_i \leftarrow j_i + 1$
		Abort if $j_i > d$
		$(\mathbf{c}, \mathbf{r}) \leftarrow (\mathbf{c}_{ij}, \mathbf{r}_{ij})$ with $j = j_i$
	$\xleftarrow{\mathbf{c}}$	
$\langle \mathbf{c} \rangle \leftarrow \mathsf{PRNG}(\mathbf{c})$		
$\widetilde{\mathbf{r}} \leftarrow \mathsf{SPUF}(\langle \mathbf{c} \rangle)$		
	$\xrightarrow{\widetilde{\mathbf{r}}}$	
		Abort if $\mathsf{FHD}(\widetilde{\mathbf{r}}, \mathbf{r}) > \tau$

Fig. 1. The basic strong PUF authentication protocol. For the SHIC version, $\widetilde{\mathbf{r}} \leftarrow \mathsf{SPUF}(\mathsf{PRNG}(\mathbf{c}))$ is replaced by $\widetilde{\mathbf{r}} \leftarrow \mathsf{SHIC}(\mathbf{c})$. Unfortunately, the latter has a high area footprint and a substantial read-out latency.

For each authentication event, a single tuple $(\mathbf{c}, \widetilde{\mathbf{r}})$ is transferred in the clear and subsequently removed from the server's list, so that the same challenge $\mathbf{c}$ is not used by the server for a future authentication event where the previously obtained response $\widetilde{\mathbf{r}}$ can be replayed by the adversary. Counter j_i on the server implements this functionality. As a result, for a strong PUF with an ideal behavior, every authentication event is unique, and the adversary with knowledge of previously exposed tuples $(\mathbf{c}, \widetilde{\mathbf{r}})$ cannot predict the response $\mathbf{r}$ for a not-yet-seen challenge $\mathbf{c}$. Further, if the same challenge is applied to a different PUF device, an unpredictable response is produced, since CRPs from one device do not reveal CRPs for another device due to chip-unique manufacturing variation. Each $\mathbf{c}_{ij}$ is derived from a *true random number generator* (TRNG) during enrollment so that it is unpredictable; else the adversary can pre-apply a challenge ahead of a genuine authentication event and spoof the response. A device identifier $\mathbf{id}$ is sent by the device so the server knows which not-yet-used challenge $\mathbf{c}$ to issue; this allows for protocol scalability. Authentication fails if the device responds with an $\widetilde{\mathbf{r}}$ that has a fractional Hamming distance (FHD) that exceeds an authentication threshold τ w.r.t. the enrolled response $\mathbf{r}$. The server initiates the protocol run to avoid a denial-of-service (DoS) via device-side-initiated $(\mathbf{c}_{ij}, \mathbf{r}_{ij})$ depletion.[1]

1. For scenarios where the device initiates, the server needs to implement extra intelligence to detect a potential DoS attack through abnormal activity monitoring and implement countermeasures such as a timeout mechanism. There is no init packet and the transaction starts with a device-initiated $\mathbf{id}$ packet.

3 Lockdown Protocol I

In the basic authentication protocol of Fig. 1, the interface is *open*: the adversary is free to perform $(\mathbf{c}, \widetilde{\mathbf{r}})$ queries, and apply an increasing number of CRPs as the training set input for a machine learning modeling attack, with challenges possibly adaptively chosen. This is problematic since the capability of the adversary in terms of available CRPs is difficult to upper-bound. In the context of a heuristic security validation against a machine learning algorithm, it is difficult to determine the size of the CRP training set and thus attack vulnerability.

To address this problem, we *lockdown* the interface as shown in Fig. 2. The server implicitly regulates the availability of CRP machine learning training material at the protocol level.

We note that the open interface protocol of Fig. 1 also faces fundamental issues: an SPUF physical circuit that can securely instantiate the protocol in Fig. 1 may not exist in practice [8]. Our practical system-level approach circumvents this circuit-level problem.

3.1 Protocol I Description

3.1.1 Enrollment

During enrollment, a one-time event that occurs at a trusted location, the server issues a deterministic challenge $\mathbf{c}$ and obtains a response that is split into $\mathbf{r}_1$ and $\mathbf{r}_2$. Specifically, from a starting challenge $\mathbf{c}$, response $\mathbf{r}_1$ is first generated. Without resetting or reseeding the challenge expansion schedule, response $\mathbf{r}_2$ is then generated. The tuple $(\mathbf{r}_1, \mathbf{r}_2)$ is stored securely for future use. This procedure is repeated d times in order to support d authentication events.

3.1.2 Authentication

During an authentication event, the server receives a device identifier $\mathbf{id}$, and then sends a packet comprising of $\mathbf{c}$ and $\mathbf{r}_1$. The device receives this information, and then compares the incoming $\mathbf{r}_1$ against a physically regenerated response $\widetilde{\mathbf{r}}_1$. If FHD is beyond a preset threshold τ, the processing aborts since the server is regarded as not authentic. Else, the remaining response bits $\widetilde{\mathbf{r}}_2$ are generated by the PUF and released outside of the device. The server completes authentication of the device by comparing $\widetilde{\mathbf{r}}_2$ against its enrolled version $\mathbf{r}_2$.

3.2 Remarks

3.2.1 Worst-Case CRP Exposure Heuristic Validation

By moving from an open interface [5], [6], [9], [10] to an interface that is locked down, we *upper-bound* the adversary's capability in terms of available CRP material. For a strong PUF, a heuristic machine learning validation based on the *worst-case* CRP training-set size can now be performed.

3.2.2 Challenge Using a Counter Value

Due to the lockdown, the adversary cannot issue a not-yet-seen packet $\mathbf{c} \parallel \mathbf{r}_1$ (not-yet-released by the server) and get the corresponding returning response packet $\widetilde{\mathbf{r}}_2$; as a result, the challenges can be *deterministically* generated, e.g., using a counter. Every authentication is unique since the challenge is non-repeating, based on a server-side counter.

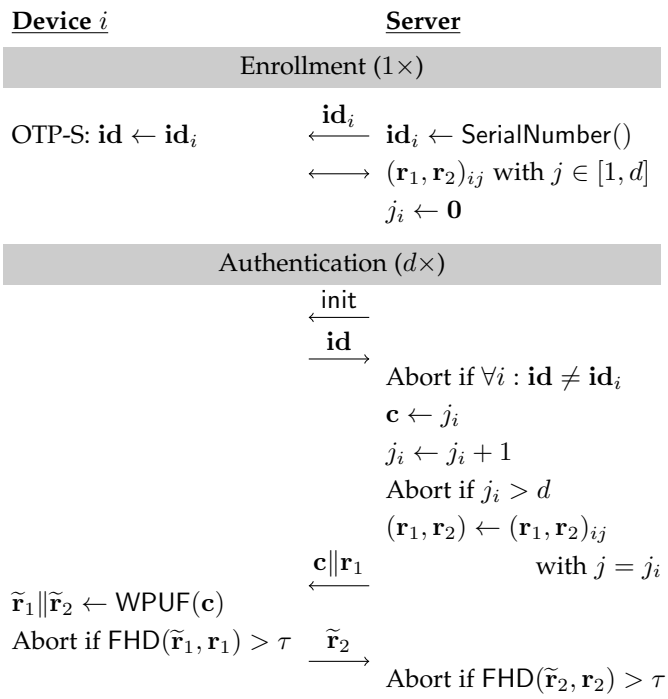


Fig. 2. Lockdown protocol I, providing unilateral authentication with a weak PUF. A strong PUF could also be used by replacing $\widetilde{\mathbf{r}}_1 \| \widetilde{\mathbf{r}}_2 \leftarrow \mathsf{WPUF}(\mathbf{c})$ with $\widetilde{\mathbf{r}}_1 \| \widetilde{\mathbf{r}}_2 \leftarrow \mathsf{SPUF}(\mathsf{PRNG}(\mathbf{c}))$.

3.2.3 Support for Weak PUFs

We allow an SRAM PUF [31] to be used for lightweight entity authentication in a manner that is secure against computationally unrestricted adversaries (cf. Sec. 9.2), assuming each SRAM cell is physically distinct and the SRAM bits are uncorrelated. The deterministic challenge counter value can be readily mapped into an SRAM memory address.

3.2.4 No Device-Side Monotonic Counter

We note that a conventional lockdown approach using a device-side monotonic counter is less compelling than our protocol-level lockdown approach in the following respects: i) there is extra device-side complexity associated with a silicon monotonic counter that requires on-chip programmable tamper-proof storage; ii) a device-side monotonic counter is more susceptible to denial-of-service attacks for an adversary with uninterrupted interface access.

3.2.5 Relationship to Mutual Authentication

We emphasize that protocol I does not perform full mutual authentication, due to a lack of device-generated "freshness". The lockdown still occurs because $\mathbf{r}_1$ and $\mathbf{r}_2$ are "locked" to each other by design; they are both generated from the same starting challenge, and challenge replay does not produce *new* response bits.

4 Lockdown Protocol II

4.1 Preventing Repeated Measurements

We introduce a protocol extension to prevent PUF noise side-channel information extraction [18], [24], [25]. We note that while protocol I in Fig. 2 prevents the adversary from

obtaining *new* CRPs without the server's permission, it does not prevent *repeated measurements* using the same challenge.

Our protocol extension requires the use of a strong PUF that supports *model-based authentication*,[2] meaning that the strong PUF circuit supports a mode of operation available only during enrollment where a linear amount of manufacturing variation information can be extracted so that an *authentication verification* model (not to be confused with an adversary's *attack* model) can be trained, and later used on the server to synthesize *any* CRP [5], [6], [32]. During enrollment, instead of storing explicit tuples (cf. Fig. 1, Fig. 2) for each device, a single authentication verification model $\widehat{\mathsf{SPUF}}_i$ is stored instead (Fig. 3).

4.2 XOR PUF with Bypass

Here, we describe a PUF circuit with a bypass mechanism where the PUF is trivially easy to learn during enrollment but not so once the device is deployed.

The original XOR arbiter PUF construction as described in [33] takes k copies of the basic arbiter PUF and merges the k output bits using bit-wise XOR, with the same challenge applied to each copy. It is well known that while the basic arbiter PUF can be learned with relative ease, the XORing produces output bits that are more difficult to learn [14]. We take advantage thereof by *making the easier-to-learn variant available during enrollment*, i.e., bypassing the XORs. Afterwards, the XORs are no longer bypassed, increasing the machine learning difficulty for the adversary.

4.3 Protocol II Description

4.3.1 Enrollment

During enrollment, instead of securely storing explicit tuples $(\mathbf{r}_1, \mathbf{r}_2)_{ij}$ as before (cf. Fig. 2), i.e., one for each authentication event supported, a single authentication verification model $\widehat{\mathsf{SPUF}}_i$ per device is stored for all future authentication events for PUF device i (Fig. 3). When implemented using an XOR PUF, the XORs are *bypassed* (cf. Sec. 4.2) so that the basic PUFs are each *individually easy to learn* [6], [23]. The enrollment interface is then disabled; we note that this requires an irreversible fuse or tamper-proof (public) storage of some sort.

4.3.2 Authentication

During an authentication event, the server obtains a device identifier $\mathbf{id}$ packet which now also includes a challenge $\mathbf{c}_D$ from the device; the device-side challenge is to allow a *challenge exchange*, so neither the device nor the server can unilaterally determine all the bits of $\langle \mathbf{c} \rangle$. The server then sends $\mathbf{c}_S$ and $\mathbf{r}_1$, the latter software-emulated using $\widehat{\mathsf{SPUF}}_i$ with $\langle \mathbf{c} \rangle$ that is a function of both $\mathbf{c}_S$ and $\mathbf{c}_D$. The device then authenticates the server by comparing the incoming $\mathbf{r}_1$ against a physically regenerated response $\widetilde{\mathbf{r}}_1$. If the fractional Hamming distance FHD is beyond τ, the process aborts since the server is regarded as not authentic. Else, the remaining response bits $\widetilde{\mathbf{r}}_2$ are generated by the PUF and released outside of the device. Finally, the server completes the authentication of the device by comparing the incoming $\widetilde{\mathbf{r}}_2$ against the $\mathbf{r}_2$ emulated using $\widehat{\mathsf{SPUF}}_i$.

2. Please refer to [6] for more details, where the scheme is referred to as parameter-based authentication.

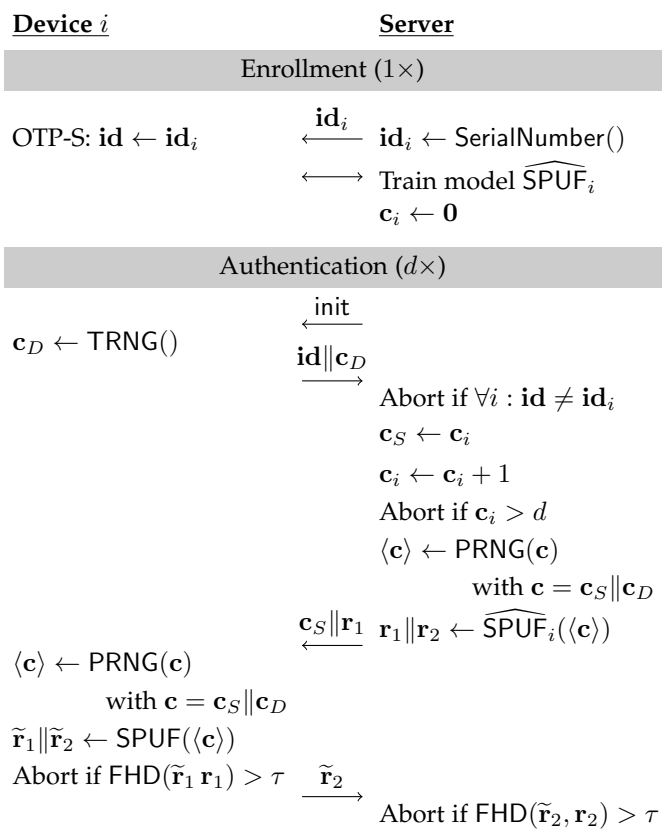


Fig. 3. Lockdown protocol II, providing mutual authentication. A device-side nonce $\mathbf{c}_D$ enables a noise side-channel countermeasure.

4.4 Remarks

4.4.1 Device-Side Nonce for "Freshness"

Recent PUF attacks used repeated measurements to i) exploit PUF noise as side-channel information [18], [24], [25]; ii) perform noise filtering [23], [26], [27]; iii) obtain backside photonic information [28], [29]. These attacks can be prevented by using a device-side nonce in the form of $\mathbf{c}_D$, similar to how nonces are used generally to prevent replay attacks. As a by-product, an added device-side nonce also enhances protocol I to support full mutual authentication.

4.4.2 Model-Based Authentication Required

For protocol II (Fig. 3), collecting explicit tuples during enrollment time is not practical since $\mathbf{c}_D$ cannot be anticipated as it is determined at run-time; this makes the use of $\widehat{\mathsf{SPUF}}_i$ a requirement. Although an enrollment disabling mechanism is needed to prevent unauthorized $\widehat{\mathsf{SPUF}}_i$ extraction once the device is deployed, there are practical advantages. The database size per device on the server is *constant*, vs. being *linear to the number of authentication events supported*. For the basic strong PUF authentication protocol (Fig. 1) as well as protocol I (Fig. 2), collecting explicit tuples is practical only in cases where the number of authentication events is relatively small. Else, server database size and enrollment time would increase dramatically.

5 PRNG Design

5.1 Design Philosophy

The PRNG is an often overlooked security bottleneck in many lightweight authentication protocols. Linear Feedback Shift Registers (LFSRs) are popular due to their low hardware footprint and nearly-uniform run statistics. Other less beneficial properties, however, are not always taken into account properly. This includes circularity, linearity, predictability and fixed points. For example, as pointed out in [32], the first reverse fuzzy extractor protocol [34] was broken due to LFSR circularity. Furthermore, both slender PUF versions [5] were broken due to LFSR linearity. Of course, the aforementioned protocols could be repaired easily via PRNG redesign.

We take it one step further. The architecture of our protocols largely facilitates the PRNG design. This substantially differs from the traditional approach in which the PRNG is considered to be an implementation matter only. In particular, counter-based server challenge $\mathbf{c}_S$ is the main benefactor. Due to its short length, i.e., $\mathsf{L}(\mathbf{c}_S) = \lceil \log_2(d) \rceil$, the adversary has restricted freedom in mounting a challenge manipulation attack.[3]

5.2 Our High-Level Design

For both protocols, the PRNG consists of a maximum-length LFSR, having state $\mathbf{s}$. The feedback polynomial, as extended over the finite field $GF(2)$, is required to be primitive. Initialized with a given seed value, the LFSR starts cycling through a subset out of $2^{\mathsf{L}(\mathbf{s})} - 1$ states. In each protocol run, the LFSR should produce $\mathsf{L}(\langle \mathbf{c} \rangle)$ bits, hereby enabling the PUF to generate $\widetilde{\mathbf{r}}_1 \| \widetilde{\mathbf{r}}_2$. For this purpose, in each cycle, the last bit of $\mathbf{s}$ is appended to the challenge stream. The all-zeros state $\mathbf{s} = \mathbf{0}$ is a fixed point, avoided by design.

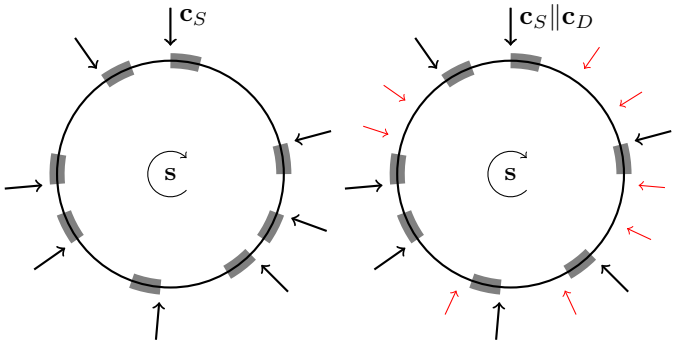
For protocol I, the seed is defined as $\mathbf{s}_0 = \mathbf{iv} \| \mathbf{c}$, and we evaluate $\mathsf{PRNG}(\mathbf{c}) = \mathsf{LFSR}(\mathbf{s}_0)$. For protocol II, the seed is defined as $\mathbf{s}_0 = \mathbf{iv} \| \mathbf{c}_S \| \mathbf{c}_D$, and $\mathsf{PRNG}(\mathbf{c}_S \| \mathbf{c}_D) = \mathsf{LFSR}(\mathbf{s}_0)$ is evaluated similarly. The initialization vector $\mathbf{iv}$ is hardcoded and identical for all fabricated devices. Figure 4 illustrates the working principles. Points via which the circular stream can be entered are indicated by an arrow. These comprise only a very small fraction of the total number of states, restricting capabilities of the adversary.

The main danger is that state segments, i.e., gray-colored arcs in Fig. 4, might overlap. An adversary that eavesdropped on prior protocol runs is hereby facilitated. Especially if the corresponding state segment offset turns out to be an integer multiple of $\mathsf{L}(\mathbf{s})$, in which case part of a previous response string $\widetilde{\mathbf{r}}_1 \| \widetilde{\mathbf{r}}_2$ could be replayed. We avoid all forms of state segment overlap.

5.3 Security Analysis for Lockdown Protocol I

We consider a protocol I scenario using a strong PUF and challenge expansion. For a given $\mathbf{iv}$, we suggest to run an *exhaustive check during the early design phase* for detecting potential overlap. Only $\mathsf{L}(\mathbf{c}) = \lceil \log_2(d) \rceil$ bits can be set. This requires $d \cdot \mathsf{L}(\langle \mathbf{c} \rangle)$ comparisons to detect re-appearance of $\mathbf{iv}$ in the course of the LFSR run. If, after the seed $\mathbf{s}_0 = \mathbf{iv} \| \mathbf{c}$ is applied, a state that begins with $\mathbf{iv}$ reappears, that particular LFSR design is disqualified. This can be remedied by an alternate $\mathbf{iv}$ value, a different feedback polynomial, etc. Candidate $\mathbf{iv}$ values can be generated arbitrarily during the early design phase, and an $\mathbf{iv}$ value that passes the exhaustive check is hard-coded into the LFSR reset value.

We impose the constraint $\mathbf{iv} \neq \mathbf{0}$, hereby disabling fixed point attacks. For a fixed point, all bits in $\widetilde{\mathbf{r}}_1 \| \widetilde{\mathbf{r}}_2$ are expected to be equal, apart from potential noisiness. The success rate for impersonating a PUF device is only $1/2$ then, which is obviously to be avoided. Furthermore, $\mathbf{iv}$ should not be instantiated with a repetitive pattern, such as alternating ones and zeros. This could induce overlap, as the exhaustive check may point out.



(a) Lockdown protocol I. (b) Lockdown protocol II.

Fig. 4. The LFSR-based PRNG of (a) lockdown protocol I and (b) lockdown protocol II. The circle represents the collection of $2^{\mathsf{L}(\mathbf{s})} - 1$ states, traversed through in clockwise direction. Gray-colored arcs represent sets of sub-challenges $\langle \mathbf{c} \rangle$ as they occur during genuine authentication runs. For protocol I, these segments are enrolled in advance by the server. For protocol II, these are determined at runtime, based on $\mathbf{c}_D$.

5.4 Security Analysis for Lockdown Protocol II

For protocol II, an exhaustive check for overlap during the design phase is no longer feasible, but we can use a *run-time check* instead. From a given seed $\mathbf{s}_0$, in the course of iterating though $\mathsf{L}(\langle \mathbf{c} \rangle)$ LFSR states, if a state that begins with $\mathbf{iv}$ reappears, the authentication process should be aborted. This run-time check can be implemented on the device by comparing $\mathsf{L}(\mathbf{iv})$ LFSR state bits against the hardwired $\mathbf{iv}$. Implementation on the server-side can be done in software.

We consider the probability that $\mathbf{iv}$ reappears under normal operations. While the LFSR has $2^{\mathsf{L}(\mathbf{s})} - 1$ states, there are only $2^{\mathsf{L}(\mathbf{s}) - \mathsf{L}(\mathbf{iv})}$ entry points. If $\mathbf{c}_D$ is randomly chosen and $\mathbf{c}_S$ uniformly chosen (it is a counter value), the probability that $\mathbf{iv}$ reappears one or more times over $\mathsf{L}(\langle \mathbf{c} \rangle)$ sub-challenges is shown in Eqn. (1). For each sub-challenge, there is a $1 - 0.5^{\mathsf{L}(\mathbf{iv})}$ probability that a re-appearance did not occur. Each authentication event uses $\mathsf{L}(\langle \mathbf{c} \rangle)$ sub-challenges. Fig. 5 shows consistency with simulated data. The equation assumes the sequence of LFSR states to be representable as a random permutation.

$$P_{iv\_reappear} = 1 - [1 - 0.5^{\mathsf{L}(\mathbf{iv})}]^{\mathsf{L}(\langle \mathbf{c} \rangle)}. \quad (1)$$

With an active adversary, $\mathbf{c}_D$ *may* be chosen adaptively while aiming to impersonate a device. The same holds for

3. For protocol I, $\mathbf{c}_S$ is just $\mathbf{c}$

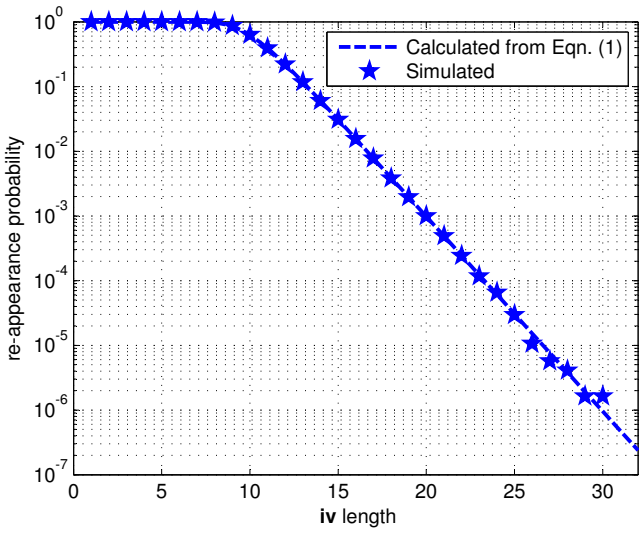


Fig. 5. Probability of initial value re-appearance. Eqn. (1) matches closely with simulated data points, obtained using a 256-bit LFSR, $\mathsf{L}(\langle \mathbf{c} \rangle)$ = 1024, and 1.2M simulation runs.

$\mathbf{c}_S$ while aiming to impersonate the server. The probability that $\mathbf{iv}$ reappears might hence be more than what is given in Eqn. (1). However, these events can be 100% reliably detected if a simple run-time check is implemented. Eqn. (1) may be viewed as a "reliability" measure under normal operations without malicious challenge modifications, and reflects the probability that an authentication retry is needed.

6 Protocol Instantiation

Several protocol instantiation values need to be selected in order to realize our scheme. This includes $\mathsf{L}(\mathbf{c}_S)$, $\mathsf{L}(\mathbf{r}_1)$, $\mathsf{L}(\mathbf{r}_2)$, $\mathsf{L}(\mathbf{iv})$ and τ for protocol I and additionally $\mathsf{L}(\mathbf{c}_D)$ for protocol II. We discuss the general properties and specify the selection procedures for a given set of constraints.

6.1 Constraints

6.1.1 Reliability

For a given use case, a target failure rate P_{fail} is imposed on each failure mechanism. This is the probability that a genuine protocol run fails due to excessive PUF noisiness, or in the case of protocol II, also due to $\mathbf{iv}$ re-appearance (cf. Eqn. 1) for a randomly generated $\mathbf{c}_D$ and a counter-based $\mathbf{c}_S$ value. In our example, we assign the same P_{fail} to both failure mechanisms. A typical specification thereof is 10^{-6}, in line with failure (retry) rates of human biometrics applications, or 10^{-9}, in line with failure rates of silicon devices fabricated using state-of-the-art process nodes. The average bit error rate of PUF response bits is denoted as $\mathsf{P_e}$. It is strongly influenced by environmental conditions tied to a specific use case.

6.1.2 Number of Genuine Authentications

The number d of genuine authentication events[4] supported is specified based on a given use case, e.g., 10 authentications for supply chain tracking where there are no more than 10 supply chain checkpoints. Alternatively, 1,000 authentications for a disposable NFC wristband for a multi-day conference. The number d is set high enough such that depletion almost never occurs. In certain use cases such as multi-day conference, in the exceptional event that depletion does occur, a new band containing a different PUF device can be issued. The system is designed such that the adversary cannot obtain CRPs associated with more than d authentication events per PUF device; with lockdown, this is possible even for an adversary that can make a very large number of queries since availability of $\mathbf{r}_1$ is implicitly regulated by the server.

4. Authentications made by the server/verifier, not by the adversary. The latter is allowed to make more than d queries.

6.1.3 Security

Finally, the adversary's success probability P_{adv} is assigned to different events. In our scheme, the adversary can breach the threshold comparison in the "reverse" direction, e.g., by guessing $\mathbf{r}_1$ to be within τ of $\widetilde{\mathbf{r}}_1$ in terms of fractional Hamming distance; or in the "forward" direction, e.g., by guessing $\widetilde{\mathbf{r}}_2$ to be within τ of $\mathbf{r}_2$. We consider the system to be compromised if authentication in *either* direction is breached, i.e., there is a false acceptance in either direction. The adversary can also repeatedly query the device to cause a $\mathbf{c}_D$ from a prior genuine authentication to reappear, in a replay attack. We assign the same P_{adv} to all these events.

A typical specification of 10^{-9} more than covers applications that require human biometrics false acceptance rates; alternatively, 10^{-20} and 10^{-40} targets 64-bit and 128-bit security levels respectively. When a strong PUF is used with protocol II, P_{adv} is influenced by the machine learning classification error. A classification error $\epsilon = 0$ means that a particular learning algorithm run can predict the PUF response with zero error, i.e., complete certainty. An $\epsilon = 0.5$ means that a particular learning algorithm run cannot predict the PUF response better than random guessing. We choose a conservative d value so that even with a CRP training-set size associated with $d-1$ authentication events, the machine learning classification error rate remains close to the ideal value of 0.5 based on a heuristic validation using today's best machine learning algorithms. In particular, we want to lower d such that the distribution of ϵ from multiple machine learning runs (with the worst case CRP training-set size) looks similar to a classification error distribution obtained through random guessing. When a properly implemented weak PUF, which uses a physically distinct component to derive each response bit, is used with protocol I, there is no machine learning attack uncertainty.

6.2 Response Lengths $\mathsf{L}(\mathbf{r}_1)$ and $\mathsf{L}(\mathbf{r}_2)$

6.2.1 False Rejection Rate

Threshold τ is applied to responses $\widetilde{\mathbf{r}}_1$ and $\widetilde{\mathbf{r}}_2$, each of which may induce a failure due to PUF noisiness. The corresponding probability is often referred to as the *false rejection rate*. In our example, we use $\mathsf{L}(\mathbf{r}_1) = \mathsf{L}(\mathbf{r}_2)$ and the same threshold τ for both, which simplifies the equations. We note that the equations can be adapted for different response lengths or thresholds in the reverse and forward directions, which we do not explicitly consider here. $\mathsf{F_{bino}}$ denotes the *cumulative distribution function* of a binomial distribution. When each direction is treated in isolation, the probability that the noise

$\mathsf{P_e}$ does not induce bit flips that exceed the threshold τ over $\mathsf{L}(\mathbf{r}_1)$ bits is given by

$$\mathsf{F_{bino}}\Big(\lfloor \tau \cdot \mathsf{L}(\mathbf{r}_1)\rfloor; \mathsf{L}(\mathbf{r}_1), \mathsf{P_e}\Big) = \sum_{b\,=\,0}^{\lfloor \tau \cdot \mathsf{L}(\mathbf{r}_1)\rfloor} \binom{\mathsf{L}(\mathbf{r}_1)}{b} \cdot (\mathsf{P_e})^b \cdot (1-\mathsf{P_e})^{\mathsf{L}(\mathbf{r}_1)-b}. \quad (2)$$

The false rejection rate accounting for both the reverse and forward directions is given below based on the union bound. The square term in the equation accounts for the fact that if either direction fails, e.g., due to excessive PUF noise, the protocol run is considered to have failed.

$$P_{fail} = 1 - \Big(\mathsf{F_{bino}}\Big(\lfloor \tau \cdot \mathsf{L}(\mathbf{r}_1)\rfloor; \mathsf{L}(\mathbf{r}_1), \mathsf{P_e}\Big)\Big)^2. \quad (3)$$

6.2.2 False Acceptance Rate

Furthermore, guessing attacks should be excluded. We assume the adversary knows the systematic bias β of the PUF population, i.e., the averaged probability that a given response bit evaluates to 1. The probability that the adversary guesses a particular "1" bit correctly is β^2, and for "0" bit it is $(1-\beta)^2$, thus the $1-(\beta^2+(1-\beta)^2) = 2\beta(1-\beta)$ argument in the *false acceptance rate* computation in Eqn. (4). We want the false acceptance rate in either direction to be sufficiently low. In our example, we assign both to P_{adv}. In PUF designs with a fully symmetrical layout, $\beta \approx 0.5$ can typically be achieved [35].

$$P_{adv} = P_{guess} = \mathsf{F_{bino}}\Big(\lceil \tau \cdot \mathsf{L}(\mathbf{r}_1)\rceil; \mathsf{L}(\mathbf{r}_1), 2\beta(1-\beta)\Big). \quad (4)$$

This equation gives the probability of the adversary guessing $\mathsf{L}(\mathbf{r}_1)$ response bits to be within threshold τ, where $2\beta(1-\beta)$ represents the success probability of guessing a single response bit correctly given the systematic bias β.

If a strong PUF is used with protocol II, the effects of machine learning can be accounted for in Eqn. (4) by substituting $2\beta(1-\beta)$ with ϵ (cf. Sec. 6.1.3).

6.3 Challenge Lengths $\mathsf{L}(\mathbf{c}_S)$ and $\mathsf{L}(\mathbf{c}_D)$

The server challenge is dimensioned based on the number of genuine authentications supported, i.e., $\mathsf{L}(\mathbf{c}_S) = \lceil \log_2(d)\rceil$. For protocol II, there is also a device challenge $\mathbf{c}_D$. Its length is chosen to counteract server impersonation via replay. In particular, the adversary that eavesdropped on $d-1$ genuine protocol runs may subsequently query the device until a previously used $\mathbf{c}_D$ appears, and then maliciously apply the corresponding $\mathbf{c}_S$ and $\mathbf{r}_1$. The probability that a $\mathbf{c}_D$ value generated from an ideal on-chip TRNG overlaps with a particular single past value is $2^{-\mathsf{L}(\mathbf{c}_D)}$. There are a maximum of $d-1$ prior $\mathbf{c}_D$ values that were used for prior genuine authentication runs. Eqn. (5) expresses the success probability of the above replay attack.

$$P_{adv} = P_{replay} = 2^{-\mathsf{L}(\mathbf{c}_D)} \cdot (d-1). \quad (5)$$

In the event of a successful replay, a previously used set of sub-challenges $\langle \mathbf{c} \rangle$ would be repeated on a malicious run, allowing for noise side-channel information extraction for potentially $\mathsf{L}(\mathbf{r}_2)$ bits.

6.4 Initialization Vector Length $\mathsf{L}(\mathbf{iv})$

For protocol I, the length of the initialization vector $\mathbf{iv}$ may be chosen arbitrarily, given that the exhaustive check excludes overlap. For protocol II, we make use of Eqn. (1), i.e., let $P_{iv\_reappear} = P_{fail}$ and solve for $\mathsf{L}(\mathbf{iv})$ for a particular choice of $\mathsf{L}(\langle \mathbf{c} \rangle)$.

7 Experimental Validation

7.1 Proof-of-Concept PUF ASIC Implementation

Our proof-of-concept implementation targets a lightweight RFID/NFC use case, and was manufactured with a low-cost $0.18\mu m$ silicon fabrication process. As shown in Fig. 6, a silicon die contains four chains each comprising of a basic 64-stage PUF (k = 4, n = 64); there is a configurable m value (using a combination of fuse and software programming) for serially merging groups of m 4-XORed response bits to form a composite output response bit $\widetilde{r}$. Each composite response bit is a function of the manufacturing variation of the four 64-stage cross-bar chains, as well as an input challenge bit applied to each one of the stages that is fed from a challenge schedule output denoted as $\langle \mathbf{c} \rangle$. In our design, each 64-stage chain receives different challenge bits, in contrast with the original XOR approach of [33] that applied the same 64-bit challenge to each chain. In terms of PUF area, each 64-stage structure contains 64 cross-bar multiplexer stages, which requires 260 NAND-2 gate equivalent (GE), with an arbiter at the end of the last stage formed using a cross-coupled NAND latch. In particular, our silicon design used a compact 4 GE for each of the 64 cross-bar multiplexer stage, and a 4 GE estimate accounts an arbiter latch. With four 64-stage chains, there are 1040 GEs total for the PUF manufacturing variation circuit.

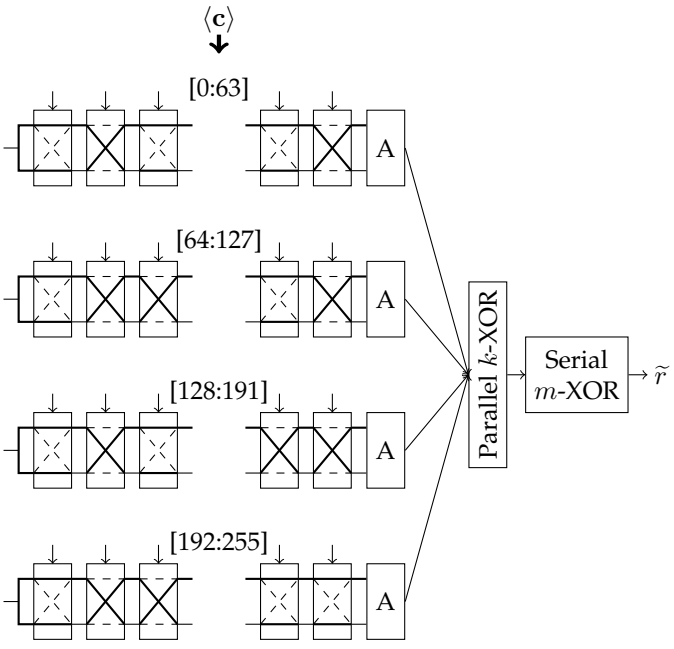


Fig. 6. XOR arbiter PUF silicon implementation.

Our implementation also has XOR bypass support (not explicitly shown in the figure) so that the authentication verification model $\widehat{\mathsf{SPUF}}_i$ can be easily derived during

enrollment time, allowing the server to later predict the response to any challenge.

7.2 Viability of Model-Based Authentication

We validated the model training approach for enrollment as was shown in Fig. 3, where $\widehat{\text{SPUF}}_i$ is stored on the server for each device instead of storing explicit tuples. During enrollment, the XORs are bypassed, so that the model for each basic 64-stage PUF can be easily derived.

We achieved an average noise level for a basic 64-stage PUF of 4.3% when the authentication is performed using a model that was learned with no more than 8192 response bits during enrollment, and where the PUF model resolution for each modeling stage used a floating point (full-resolution) representation. Using a decimated 7-bit representation, the average noise level for a basic 64-stage PUF increased marginally to 4.5%. These noise levels are for an operating temperature span of -25°C to 85°C ambient, and supporting enrollment nominally at 25°C.

We note that the noise level for a basic arbiter PUF implementation in [35] was shown to be 4% for an environmental variation between -40°C and 85°C. We achieved similar noise levels *while* validating against a model (of the basic 64-stage PUF) derived using machine learning techniques.

7.3 Heuristic Evaluation: Protecting Against Today's Best Machine Learning Attacks

Fig. 7 shows the best learning attack results for the k-XOR arbiter PUF to date that is pertinent to our discussion.

- **Line width** indicates whether *noise side-channel information* is exploited (thin line) or not (thick line).
- **Line dash-style** indicates either *same* challenges are applied to each of the k PUF chains (dashed line) or *different* challenges (solid line).

7.3.1 *Ruhr-University Bochum Attacks (2015)*

The most recent results are from Ruhr-University Bochum, shown in red (circles). These include the best attack results for XOR PUFs to date, which exploits noise side-channel (s.c.) information to perform reliability-based machine learning [18], as shown by red-*thin* lines. This class of s.c. attack can be eliminated by preventing repeated challenges [18]. If so, the red-thin results are no longer relevant, and the Bochum red-*thick* results now apply [17].

It is possible to create a heuristically-secure lightweight PUF authentication scheme against these recent Bochum attacks through a lockdown of CRP exposure. If protocol I is used (no countermeasure applied), a k=8-XOR n=128-stage PUF requires 300,000 CRPs to break using the red-thin results from [18]. In Fig. 7, the x-axis represents k, the number of n-stage PUFs whose output response bits are XORed together, and y-axis represents number of CRPs where each response is a single bit. In this case, $R^{limit} < 300,000$. If 1,000 bits are used per authentication for each of $\mathbf{r}_1$ and $\mathbf{r}_2$, $d = R^{limit}/(\mathsf{L}(\mathbf{r}_1)+\mathsf{L}(\mathbf{r}_2)) < 150$ can be supported. A more exact number would require more experiment runs, to lower the CRPs until the machine learning algorithm has trouble converging.

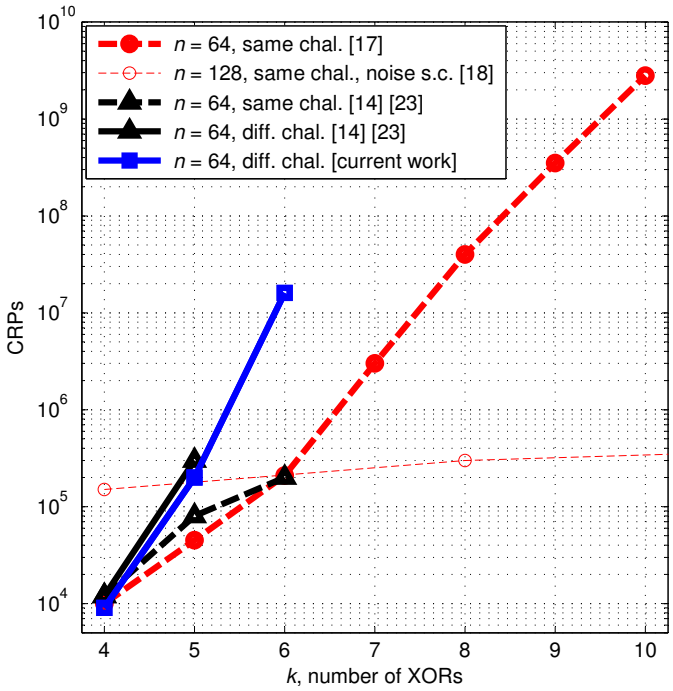


Fig. 7. Heuristic machine learning results for k-XOR PUF. Red (circles) are from Ruhr-University Bochum (2015), and black (triangles) are from Technical University Munich (2010-2014). The most difficult XOR PUF configuration to attack is one i) without benefit of noise side-channel (s.c.) information; and ii) using different challenges for each of the k PUF chains. We confirmed and extended these results in blue (squares).

Now we analyze the Bochum results with respect to our protocol II, which contains a noise s.c. attack countermeasure. The results that apply are no longer those from [18] (which requires noise s.c. information), but the ones in [17], which are shown as red-thick lines. For an 8-XOR PUF, $4 \cdot 10^7$ CRPs were used, which supports $1 \cdot 10^4$ authentication events. For 10-XOR PUF, $3 \cdot 10^9$ CRPs are required for the attack, which supports $1 \cdot 10^6$ authentication events.

7.3.2 *Technical University Munich Attacks (2010, 2013)*

Shown in black (triangles) in Fig. 7 are several recent attacks by Technical University Munich [14], [23]. The black-*solid* line indicates results for "lightweight" PUFs, which corresponds to applying *different* challenges to each of the k PUF chains. The black-*dashed* line represents *same* challenges applied to each chain, which is easier to attack (requires less CRPs to learn, given the same n and same k). We note that the *dashed thick* results from Munich and Bochum look consistent, representing attacks without the benefit of noise s.c. information and with same challenges for each chain.

7.3.3 *Our Confirmation Attacks*

In our case, we want to find an XOR PUF configuration that the data have shown to be the *most difficult to attack*, and construct / hardwire our PUF instantiation. From Fig. 7, we make the following observations.

i) Without the benefit of noise s.c. information, the number of CRPs required to learn a k-XOR PUF is no longer flat (thin line) but grows with k (thick lines) – *achievable using a noise s.c. countermeasure* (cf. Sec. 4).

TABLE 1
m-way serial XOR merge (using k=4-XOR n=64-stage PUF)

	m=4 (from ([18]) (uses noise s.c.)	m=2 (with countermeasure)	m=4 (with countermeasure)
CRPs	40,000	$\geq$ 12,000,000	>> 12,000,000

ii) We can get an additional improvement by using different challenges per chain (solid lines) vs. same challenges (dashed lines), the former having a steeper slope in Fig. 7 – *achievable by not replicating PRNG bits across multiple chains* (i.e., avoid the original XOR PUF [33]).

Our confirmation attacks are shown in blue (square), where we obtained successful modeling for 4-XOR at 9000 CRPs, 5-XOR at 200,000 CRPs. This is consistent with the results by the Munich group (solid black). We then proceeded to lower the CRPs until the machine learning algorithm had trouble converging.[5] We were unable to model a 4-XOR at 8500 CRPs, a 5-XOR at 100,000 CRPs, and a 6-XOR at 16,000,000 CRPs, after over 100 days of attack in aggregate. On a logarithmic scale, the difference between the "successful attack" point and "unable to model" point is very small. We use the RPROP algorithm [14], [16], which has been the most effective algorithm to date to attack an k-XOR n-stage PUF with no noise s.c. information.

We also performed attacks where we serially merged m k-XOR results. Table 1 shows results for k=4, n=64, and compares against results from [18], the latter taking advantage of noise side-channel information. All results here are for different challenges applied to each chain. Reliability-based machine learning in [18] successfully attacked a k=4-XOR PUF with m=4 using 40,000 CRPs. Our experiment to date with only m=2 and a countermeasure shows that using 12,000,000 CRPs, the m=2 case cannot be modeled after 100+ days. If m=4, much greater than 12,000,000 CRPs can be supported, with a more precise (higher) CRP number to be determined based on future results.

7.4 Remarks

We note that without CRP lockdown, all the attack points shown in Fig. 7 and Table 1 are likely to be broken from an analytical attack standpoint, since even in a few days or weeks time it may be feasible to obtain millions of CRPs. With lockdown, the server implicitly restricts the CRP exposure such that authentication can be performed.

8 Envisioned Architecture

8.1 Strong PUF Instantiation

Fig. 8 shows an envisioned lockdown architecture using a strong PUF, e.g., the XOR PUF in Fig. 6. We use an LFSR design that detects re-appearance of the $\mathbf{iv}$ at run-time, which prevents response overlaps with 100% certainty. Using the XOR PUF in Fig. 6 as an example, which has challenge input bits of [0:63], [64:127], [128:191], and [192:255] for the four

5. E.g., with the machine learning classification error ϵ vacillating around 0.50 with a normal distribution.

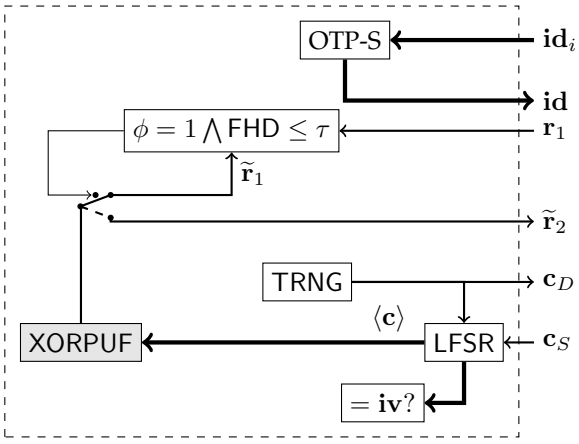


Fig. 8. Envisioned architecture for a strong PUF instantiation, using an XOR PUF (cf. Fig. 6) and lockdown protocol II. Multi-bit and single-bit data flows are drawn thick and thin respectively. On-chip storage for $\mathbf{id}_i$ only needs to be one-time programmable.

64-stage cross-bar chains, a 256-bit LFSR provides the necessary challenge expansion. This LFSR can have $\mathsf{L}(\mathbf{c}_S)$ = 20, $\mathsf{L}(\mathbf{c}_D)$ = 172, and $\mathsf{L}(\mathbf{iv})$ = 64, where the initial value $\mathbf{iv}$ is present in the LFSR reset state. To detect re-appearance of the initial value, $\mathsf{L}(\mathbf{iv})$ bits of the LFSR corresponding to the $\mathbf{iv}$ can be compared with a hardwired comparator comprising of a series of AND gates tapping either the true or the complement output of the LFSR flip-flops.

The authentication occurs in two phases. During the "reverse" authentication phase, i.e., ϕ = 0, the incoming $\mathbf{r}_1$ is compared against the physically regenerated $\widetilde{\mathbf{r}}_1$ from the PUF. Only if the comparison passes the τ threshold test are the remaining PUF response bits $\widetilde{\mathbf{r}}_2$ released outside the device during the "forward" authentication phase, i.e., ϕ = 1. This functionality is captured by the switch in the figure where the bottom path is taken only if the control condition is true. This instantiation uses lockdown protocol II to prevent noise side-channel information extraction and other attacks that require repeated measurements. A TRNG allows the device to generate a part of the starting challenge.

8.2 Weak PUF Instantiation

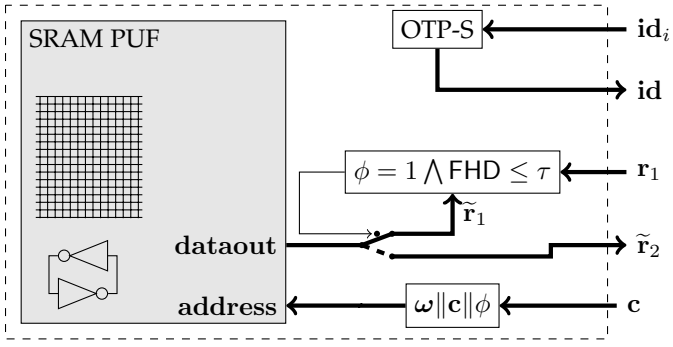


Fig. 9. Envisioned architecture for a weak PUF instantiation, using an SRAM PUF and lockdown protocol I.

Fig. 9 shows the envisioned architecture for a weak PUF instantiation, using an SRAM PUF [31]. By using lockdown protocol I, *the adversary can no longer arbitrarily*

TABLE 2
Lockdown with SRAM PUF and with XOR PUF. Supporting up to d=1,000 authentications. False acceptance rate $\approx 2^{-128}$. False rejection rate $\leq 10^{-6}$

	SRAM PUF	XOR PUF
Noise (using [35])	7.0%	4.3%
Response Size	$\mathsf{L}(\mathbf{r}_1) = \mathsf{L}(\mathbf{r}_2)$ = 320	$\mathsf{L}(\mathbf{r}_1) = \mathsf{L}(\mathbf{r}_2)$ = 1280
Threshold	$\tau = \frac{48}{320}$	$\tau = \frac{407}{1280}$
Manufacturing Variation Unit	6 Transistor SRAM Cell (1.5 GE)	16 Stacked Transistor Delay Stage (4 GE)
Size for d=1,000 authentication	640,000 6T Cells (960,000 GE) from 80 KByte SRAM	$4 \cdot 64$ delay stages (1,024 GE)
If only d=10 authentications	6,400 6T Cells (9,600 GE) from 800 Byte SRAM	$4 \cdot 64$ delay stages (1,024 GE)
Protocol	Lockdown Protocol I	Lockdown Protocol II
Security w.r.t. Learning	Secure Against *Any* Learner	Heuristic
CRPs exposable	$= d \cdot 320 \cdot 2$	$\geq 12,000,000$ (see Table 1)

read the memory contents (corresponding to the response bits), thereby addressing practical limitations associated with the SHIC approach; an extremely large memory array with an extremely slow read-out rates is no longer necessary to prevent exhaustive read-out by the adversary. We also preserve the main security property of SHIC, which is secure against computationally unrestricted adversaries, but do so for a practically instantiable SRAM PUF.

The SRAM PUF comprises of cross-coupled inverters and two access transistors, corresponding to a 6T (six transistor) memory cell, whose initial power-up value is subject to manufacturing variation and is used as a PUF response bit. There is address decode logic to determine which memory address word to output at the dataout port. To support protocol I, the memory space is divided into even entries, and odd entries, controlled by ϕ. A $\phi = 0$ indicates that the "reverse" authentication phase is active, and $\phi = 1$ indicates that the "forward" authentication phase is active. The device outputs the SRAM PUF contents for the odd addresses (for the "forward" authentication) only if the "reverse" authentication succeeds, i.e., the incoming $\mathbf{r}_1$ is within τ of $\widetilde{\mathbf{r}}_1$ in terms of fractional Hamming distance. In terms of address generation, the least significant bit is ϕ, followed by the incoming challenge $\mathbf{c}$ in the middle bits, followed by the word address $\boldsymbol{\omega}$ in the upper bits; the response length, e.g. $\mathsf{L}(\mathbf{r}_1)$, is a multiple of the memory word size (e.g., 32 bits) in this architecture.

8.3 Comparison Against Strong PUF Instantiation

Table 2 compares the weak PUF lockdown instantiation of the prior section against a strong PUF instantiation to support d = 1,000 authentications; the latter instantiation uses protocol II with k=4-XOR and m=2. To normalize comparison, we use noise results from [35] that covers a temperature range from -40°C to 85°C, with enrolled reference at 25°C. The memory PUF noise was 7.0% and arbiter PUF was 4.0%. For loss associated with using model-derived values (vs. explicit tuples) as reference, we add 0.3% more noise (cf. Sec. 7.2). From the noise level, we derive response size and threshold value for achieving a false acceptance rate $\approx 2^{-128}$ and false rejection rate below ppm (part-per-million) levels using both Eqns. (4) and (3). Next, the NAND-2 Gate-Equivalent (GE) area for the basic manufacturing variation unit is described (1 NAND2 = 4 transistors). Following that, size of the manufacturing variation portion of circuit for d=1,000 authentications is shown; for comparison, GE for d = 10 authentications is also shown.

Note: The k-XOR construction has a noise level for the false rejection rate computation that grows with $k \cdot m$, which is the total number of bits XORed together. This is similar to the "k-XOR noise" equation in [6], where the binomial distribution is used. This usage is justifiable assuming that the PUF noise affects each of the k response bits from their respective PUF chain in an independent fashion. This holds true if each PUF chain is physically distinct, and random challenges are applied to each chain (e.g., from a PRNG). We adapt the equation for the $k \cdot m$ case. If the noise level for a single n-stage PUF is $\mathsf{P_e}(1)$, then the extrapolated noise level for $k \cdot m$ bits XORed together is[6]

$$\mathsf{P_e}(k{\cdot}m) = 1 - \sum_{b\,=\,0}^{\lfloor \frac{k\cdot m}{2} \rfloor} \binom{k \cdot m}{2b} \cdot \mathsf{P_e}(1)^{2b} \cdot (1 - \mathsf{P_e}(1))^{k\cdot m - 2b}. \tag{6}$$

When $k \cdot m$ bits are XORed together, we get the *correct* result when all $k \cdot m$ bits match (between the enrollment and a later authentication query), or when an *even* number of bits mismatch. $\mathsf{P_e}(1)$ is the 4.3% value for arbiter PUF. At m=1 and k=4, the extrapolated post-XOR noise level is $\mathsf{P_e}(4 \cdot 1)$ = 0.1511 (vs. 0.1499 from silicon); at m = 2, extrapolated $\mathsf{P_e}(4 \cdot 2)$ = 0.2565 (vs. 0.2548 from silicon). The latter extrapolation is used as $\mathsf{P_e}$ in Eqn. (3) to compute the response length to achieve a false rejection rate below 1 part per million while also satisfying the false acceptance rate requirement.

9 Comparisons

9.1 Against Survey Results From [4]

Table 3 compares our approach against the finalists of a recent survey on PUF-based authentication protocols [4]. Most of these require either a cryptographic algorithm or an error correction code.[7] For example, a *controlled PUF* [2] applies hashing to a PUF's challenge/response behavior, which requires error correction since the noisy response bits are hashed. In terms of implementation complexity, our approach has a relative advantage in that we do not require on-chip error correction / cryptography; we also do not need to be concerned about the handling of error correction helper data in terms of where to store them and how to protect their integrity to prevent helper data manipulation attacks [38]. However, our lockdown approach requires the number of genuine authentication events to be limited based on a given use case.

6. $\mathsf{P_e}(1)$ denotes noise level without XORs, and $\mathsf{P_e}(k \cdot m)$ is the noise level for $k \cdot m$ PUF bits bit-wise XORed together.

7. An example PUF with error correction requires 5,200 or more GE [36], and an AES block cipher requires an additional 2,400 GE [37].

TABLE 3
Comparing *Lockdown* with protocol finalists from [4] and SHIC from [7]. *Lockdown Ia* uses a strong PUF, and *Lockdown Ib* a weak PUF. *Lockdown IIa* uses a PAC-learnable configuration but with restricted CRP exposure. *Lockdown IIb* uses a non-PAC-learnable [11] configuration, with CRP exposure also restricted.

.

Protocol	Crypto algorithm	Error correction	TRNG	Exhaustive	Mutual auth.	Number of auth.	Scalable	Privacy	PUF-independent	Modeling robust
	Heavyweight									
Ref. II-A	✓	✓	×	×	×	∞	✓	×	✓	✓
Sadeghi	✓	✓	✓	×	×	∞	×	✓	✓	✓
Controlled	✓	✓	×	×	×	∞	✓	×	×	✓
Rever. FE I	✓	∼	✓	×	✓	∞	✓	×	×	✓
Rever. FE II	✓	∼	✓	×	✓	∞	✓	×	✓	✓
SHIC	×	×	×	✓	×	∞	✓	×	×	✓
	Lightweight									
Slender	×	×	✓	×	×	∞	✓	×	×	∼
Noise bifur.	×	×	✓	×	×	∞	✓	×	×	∼
Lockdown Ia	×	×	×	×	×	d	✓	×		∼
Lockdown Ib	×	×	×	×	×	d	✓	×	✓	✓
Lockdown IIa	×	×	✓	×	✓	d	✓	×		∼
Lockdown IIb	×	×	✓	×	✓	d	✓	×		✓

Among the surveyed finalists, only the *slender PUF* [5] and *noise bifurcation* [6] protocols require neither an error-correction code nor a cryptographic algorithm. However, their PRNG, a critical lightweight building block, was respectively broken and unspecified; we made advances in this area (cf. Sec. 5). More significantly, both have an open interface, allowing the adversary to arbitrarily obtain information associated with *new* CRPs so long as access to the device's interface is maintained. With lockdown, the adversary can no longer obtain a new response $\widetilde{\mathbf{r}}_2$ without already knowing $\mathbf{r}_1$, and the release of $\mathbf{r}_1$ is implicitly enforced by the server. There is, however, a small incremental overhead for maintaining a counter state for each device on the *server*-side.[8] In exchange for this incremental complexity on the server, we *upper bound* the power of the adaptive chosen-challenge machine-learning-equipped adversary in terms of the *worst-case CRP training-set size*. Our lockdown scheme can be also viewed as a generic security notion that can complement prior approaches such as slender PUF [5] so that a heuristic machine-learning-based validation based on the *worst-case CRP exposure* becomes feasible to perform. Slender PUF also requires server-side storage of an authentication verification model for each PUF device, so only an additional server-side counter state needs to be added for each slender device in order to incorporate lockdown.

Our lockdown approach also exhibits PUF independence (works with both strong and weak PUFs). In fact, our SRAM instantiation is the first practically realizable lightweight authentication scheme that is secure against *computationally unrestricted* adversaries (cf. Sec. 9.2), which includes *any* learner; this is a strong security guarantee not possible with any of the nineteen surveyed approaches in [4].

8. There is no counter state on the *device*-side, which would be more complex to implement. E.g., requiring an on-chip monotonic counter.

9.2 Against Super High Information Content (SHIC)

The main security property of SHIC (cf. Sec. 2.3) is that it is secure against computationally unrestricted adversaries. We achieve that goal but do not require the "brute-force" SHIC approach of using an extremely large memory and extremely slow read-out speeds (cf. Sec. 8.2), and can use SRAM vs. experimental memory.

9.3 Against Theoretical Machine Learning Results

According to recent theoretical machine learning results from Ganji et al. [11], certain practically realizable k-XOR PUF constructs were mathematically shown to be learnable under the celebrated PAC framework [12], which links learning with computational complexity theory: these constructs can be learned with a polynomial number of CRPs and in polynomial time. Notably, polynomial in PUF circuit size. This same work also declared certain k-XOR PUF constructs as not PAC-learnable; they cannot be learned with a polynomial number of CRPs and in polynomial time, provided that the Vapnik-Chervonenkis (VC) dimension [39] of a k-XOR n-stage PUF is linear to $(n+1)^k$. The results from [11] assume that there are no repeated challenges. The challenge-exchange from our protocol II satisfies this condition. To date, there are no machine learning representations of a k-XOR n-stage PUF with no repeated challenges where the representation is logarithmic to $(n+1)^k$.

Recall that k is the number of PUF chains, n is the number of stages per chain, and $k \cdot n$ is the circuit size. According to [11], if the number of chains is much greater than the natural log of the number of stages per chain, such an XOR PUF theoretically requires an exponential effort to learn; specifically, with a linear increase in circuit size (number of chains), there is an exponential increase in the attack effort in terms of both CRPs and run-time. The authors of [11] concluded that with n=64-stages, if the number of chains ≥ 5, the configuration is operating in the exponential region of the learning difficulty curve; this is consistent with the fact that for n=64, $k \geq \mathsf{ln}(64) = 4.17$. We note that this result is in agreement with the thick-lined exponential curves in Fig. 7 (y-axis reflects an exponential CRP requirement in k). However, [11] *wrongly implied that it is practically infeasible* to operate in the part of the curve where the learning difficulty is exponential in the circuit size (i.e., in k), due to additive response noise. Their result presumes that *in practice*, k cannot exceed the bound $\lceil \mathsf{ln}(n) \rceil$; and *ideally* (with practicality aside), k cannot exceed 12-XOR for a presumed noise level of 4% due to a 50% noise limit.

Our silicon PUF noise level was $\mathsf{P_e}(1)$ = 4.3% (cf. Sec. 7.2). Unlike what [11] was implying, the effect of XORing is not a strict arithmetic summation, and is more precisely given by Eqn. (6). For example, for 12-XOR, the noise level based on an "arithmetic sum" extrapolation is $4.3\% \cdot 12 = 51.6\%$, which [11] implied is unusable as majority voting cannot even help. Based on our more precise equation, the 12-XOR noise level is instead 33%. Fig. 10 contains a comparison of the two approaches, showing that the "arithmetic sum" extrapolation is less accurate. We further note that [11] did not explicitly consider increasing the response length to compensate for an increase in noise level due to XORing. For 12-XOR, with a response length

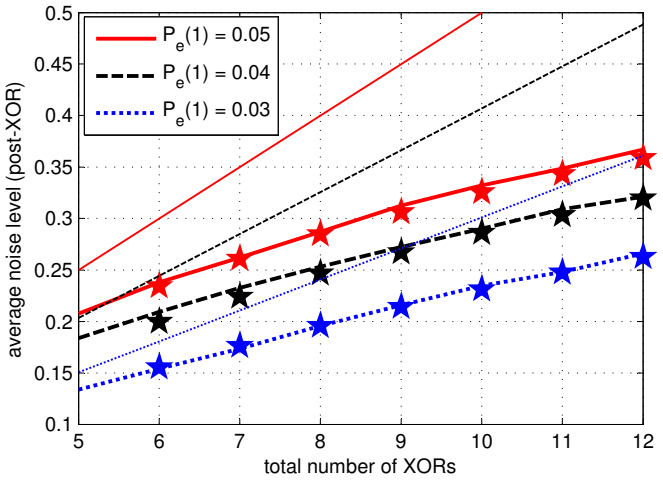


Fig. 10. XOR Noise Extrapolation: "Arithmetic Sum" vs. Eqn. (6). *Extrapolated* XOR noise for $P_e(1)$ = 0.05, 0.04, and 0.03 are shown with solid lines, dashed lines, and dotted lines, respectively, where the thin-line variant is a strict arithmetic sum extrapolation, and the thick-line variant uses Eqn. (6). The latter extrapolation more accurately matches the *simulated* noise results, shown as stars, obtained by injecting noise to each simulated PUF stage for the k chains (m = 1); k response bits are XORed to obtain the simulated post-XORed noise level.

of 3072 and a threshold of $\frac{1143}{3072}$, the false rejection and false acceptance rates of our prior examples are met. The theoretical results from [11] clearly concluded that 12-XOR is exponentially difficult to learn. We have shown that *it is in fact feasible* to operate in a region where k exceeds the bound $\lceil \ln(n) \rceil$, and even at k=12-XOR, by using a more precise noise equation, lengthening the response to preserve the false positive and false negative rates, and using our silicon data, all in the context of our protocol II. In this regime, a learning attack effort that is less than exponential (in terms of CRPs and runtime) is theoretically not possible based on the PAC learning results of [11].

10 Conclusion

Entity authentication is normally implemented with a cryptographic algorithm (e.g., AES, HMAC) and a secret key, to allow an exponential number CRPs to be derived from a linearly-sized circuit. For over a decade, there were several proposals to replicate such a behavior in a lightweight fashion using PUFs and without the aid of an error correction code or a cryptographic algorithm [4], [8], [32]. From a machine learning attack standpoint, these proposals either i) were broken; or ii) were difficult to heuristically validate on a *worst case CRP basis* due to an "open" interface.

Using our lockdown approach, we presented i) the first strong PUF scheme without error-correction where a machine learning validation run based on a *worst-case CRP training-set size* is possible; ii) first weak PUF protocol using SRAM that is secure against computationally unrestricted adversaries, including *any* learner. Finally, we are the first to confirm the existence of a PUF system level instantiation that is exponentially difficult to learn based on recent results [11] using PAC learning but is yet *practically realizable*, backed by an improved XOR noise equation, a response lengthening analysis, and our silicon data, and using our extended protocol. This final result hinges on the inability to find an k-XOR n-stage PUF representation having a VC dimension that is logarithmic to $(n+1)^k$ without using repeated challenges, which is an area of future research.

Our lockdown approach can be also viewed as a generic security notion applicable to other lightweight authentication schemes, to *upper-bound* the power of a learning adversary in terms of CRP training-set size, making security easier to justify and validate.

Acknowledgement

This work was partly funded by the Bavaria California Technology Center (BaCaTeC) through grant number 2014-1/9.

References

[1] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon Physical Random Functions," in *Proc. 9th ACM Conference on Computer and Communications Security (CCS)*, 2002, pp. 148–160.

[2] ——, "Controlled Physical Random Functions," in *Proc. 18th Annual Computer Security Applications Conference (ACSAC)*, 2002, pp. 149–160.

[3] B. Gassend, "Physical Random Functions," in *Master's Thesis, Dept. EECS, Massachusetts Institute of Technology*, 2003.

[4] J. Delvaux, R. Peeters, D. Gu, and I. Verbauwhede, "A Survey on Lightweight Entity Authentication with Strong PUFs," *ACM Computing Surveys*, vol. 48, no. 2, pp. 26:1–26:42, 2015.

[5] M. Majzoobi, M. Rostami, F. Koushanfar, D. Wallach, and S. Devadas, "Slender PUF Protocol: A Lightweight, Robust, and Secure Authentication by Substring Matching," in *Proc. IEEE Symposium on Security and Privacy Workshops*, 2012, pp. 33–44.

[6] M. Yu, D. M'Raïhi, I. Verbauwhede, and S. Devadas, "A Noise Bifurcation Architecture for Linear Additive Physical Functions," in *Proc. IEEE Int'l Symposium on Hardware-Oriented Security and Trust, HOST*, 2014, pp. 124–129.

[7] U. Rührmair, C. Jaeger, M. Bator, M. Stutzmann, P. Lugli, and G. Csaba, "Applications of High-Capacity Crossbar Memories in Cryptography," *IEEE Transactions on Nanotechnology*, vol. 10, no. 3, pp. 489–498, 2011.

[8] R. Maes, "Physically Unclonable Functions: Constructions, Properties and Applications," in *PhD Thesis, Dept. Electrical Engineering (ESAT), Katholieke Universiteit Leuven (KU Leuven)*, 2012.

[9] S. Devadas, E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal, "Design and Implementation of PUF-Based Unclonable RFID ICs for Anti-Counterfeiting and Security Applications," in *Proc. 2nd IEEE Int'l Conference on RFID*, 2008, pp. 58–64.

[10] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight Secure PUFs," in *Proc. Int'l Conference on Computer-Aided Design (ICCAD)*, 2008, pp. 670–673.

[11] F. Ganji, S. Tajik, and J.-P. Seifert, "Why Attackers Win: On the Learnability of XOR arbiter PUFs," in *Proc. Int'l Conference on Trust and Trustworhty Computing (TRUST)*, 2015, pp. 22–39.

[12] L. Valiant, "A Theory of the Learnable," *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.

[13] J. Guajardo, S. Kumar, G. J. Schrijen, and P. Tuyls, "FPGA Intrinsic PUFs and Their use for IP Protection," in *Proc. 9th Int'l Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2007, pp. 63–80.

[14] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling Attacks on Physical Unclonable Functions," in *Proc. 17th ACM Conference on Computer and Communications Security (CCS)*, 2010, pp. 237–249.

[15] C. Bishop, "Pattern Recognition and Machine Learning," *Springer*, 2007.

[16] M. Riedmiller and H. Braun, "A Direct Adaptive Method for Faster Backpropagation Learning: the RPROP Algorithm," in *Proc. IEEE Int'l Conference on Neural Networks*, 1993, pp. 586–591.

[17] J. Tobisch and G. Becker, "On the Scaling of Machine Learning Attacks on PUFs with Application to Noise Bifurcation," in *Proc. Int'l Workshop on Radio Frequency Identification, Security and Privacy Issues. (RFIDSec)*, 2015, pp. 17–31.

[18] G. Becker, "The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs," in *Proc. 17th Int'l Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2015, pp. 535–555.

[19] D. Lim, "Extracting Secret Keys from Integrated Circuits," in *Master's Thesis, Dept. EECS, Massachusetts Institute of Technology*, 2004.

[20] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing Techniques for Hardware Security," in *Proc. IEEE Int'l Test Conference (ITC)*, 2008, pp. 1–10.

[21] F. Sehnke, C. Osendorfer, J. Sölter, J. Schmidhuber, and U. Rührmair, "Policy Gradients for Cryptanalysis," in *Proc. 20th Int'l Conference on Artificial Neural Networks*, 2010, pp. 168–177.

[22] G. Hospodar, R. Maes, and I. Verbauwhede, "Machine Learning Attacks on 65nm Arbiter PUFs: Accurate Modeling Poses Strict Bounds on Usability," in *Proc. IEEE Int'l Workshop on Information Forensics and Security*, 2012, pp. 37–42.

[23] 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*, vol. 8, no. 11, pp. 1876–1891, 2013.

[24] J. Delvaux and I. Verbauwhede, "Side Channel Modeling Attacks on 65nm Arbiter PUFs Exploiting CMOS Device Noise," in *Proc. IEEE Int'l Symposium on Hardware-Oriented Security and Trust (HOST)*, 2013, pp. 137–142.

[25] ——, "Fault Injection Modeling Attacks on 65 nm Arbiter and RO Sum PUFs via Environmental Changes," *IEEE Transactions on Circuits and Systems*, vol. 61-I, no. 6, pp. 1701–1713, 2014.

[26] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, M. Majzoobi, F. Koushanfar, and W. Burleson, "Efficient Power and Timing Side Channels for Physical Unclonable Functions," in *Proc. 16th Int'l Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2014, pp. 476–492.

[27] G. Becker and R. Kumar, "Active and Passive Side-Channel Attacks on Delay Based PUF Designs," *IACR Cryptology ePrint Archive, Report 2014/287*, 2014.

[28] S. Tajik, E. Dietz, S. Frohmann, J. Seifert, D. Nedospasov, C. Helfmeier, C. Boit, and H. Dittrich, "Physical Characterization of Arbiter PUFs," in *Proc. 16th Int'l Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2014, pp. 493–509.

[29] F. Ganji, J. Krämer, J.-P. Seifert, and S. Tajik, "Lattice Basis Reduction Attack Against Physically Unclonable Functions," in *Proc. 22th ACM Conference on Computer and Communications Security (CCS)*, 2015, pp. 1070–1080.

[30] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Proceedings on the 19th Annual Int'l Cryptology Conference (CRYPTO)*, 1999, pp. 388–397.

[31] D. Holcomb, W. Burleson, and K. Fu, "Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, 2009.

[32] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Secure Lightweight Entity Authentication with Strong PUFs: Mission Impossible?" in *Proc. 16th Int'l Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2014, pp. 451–475.

[33] G. Suh and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," in *Proc. 44th Design Automation Conference (DAC)*, pp. 9–14.

[34] 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 *Proc. 16th Int'l Conference on Financial Cryptography (FC) and Data Security*, 2012, pp. 374–389.

[35] S. Katzenbeisser, Ü. Koçabas, V. Rozic, A. Sadeghi, I. Verbauwhede, and C. Wachsmann, "PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon," in *Proc. 14th Int'l Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2012, pp. 283–301.

[36] V. van der Leest, B. Preneel, and E. van der Sluis, "Soft Decision Error Correction for Compact Memory-Based PUFs Using a Single Enrollment," in *Proc. 14th Int'l Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2012, pp. 268–282.

[37] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang, "Pushing the Limits: A Very Compact and a Threshold Implementation of AES," in *Proc. 30th Annual Int'l Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2011, pp. 69–88.

[38] 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*, vol. 34, no. 6, pp. 889–902, 2015.

[39] A. Blumer, A. Ehrenfeuct, D. Haussler, and M. Warmuth, "Learnability and the Vapnik-Chervonenkis Dimension," *Journal of the ACM (JACM)*, vol. 36, no. 4, pp. 929–965, 1989.