Journal of Cryptographic Engineering (2022) 12:387–412
https://doi.org/10.1007/s13389-021-00283-6

REGULAR PAPER

Secret-free security: a survey and tutorial

Ulrich Rührmair[1,2]

Received: 30 June 2021 / Accepted: 24 December 2021 / Published online: 2 March 2022

Abstract
"Classical keys," i.e., secret keys stored permanently in digital form in nonvolatile memory, appear indispensable in modern computer security—but also constitute an obvious attack target in any hardware containing them. This contradiction has led to perpetual battle between key extractors and key protectors over the decades. It is long known that physical unclonable functions (PUFs) can at least partially overcome this issue, since they enable secure hardware without the above classical keys. Unfortunately, recent research revealed that many standard PUFs still contain other types of "secrets" deeper in their physical structure, whose disclosure to adversaries breaks security as well: Examples include the manufacturing variations in SRAM PUFs, the power-up states of SRAM PUFs, or the signal delays in Arbiter PUFs. Most of these secrets have already been extracted in viable attacks in the past, breaking PUF-security in practice. A second generation of physical security primitives now shows potential to resolve this remaining problem, however. In certain applications, so-called Complex PUFs, SIMPLs/PPUFs, and UNOs are able to realize not just hardware that is free of classical keys in the above sense, but completely "secret-free" instead. In the resulting hardware systems, adversaries could hypothetically be allowed to inspect every bit and every atom, and learn any information present in any form in the system, *without* being able to break security. Secret-free hardware would hence promise to be innately and permanently immune against any physical or malware-based key-extraction: There simply is no security–critical information to extract anymore. Our survey and tutorial paper takes the described situation as starting point, and categorizes, formalizes, and overviews the recently evolving area of *secret-free security*. We propose the attempt of making hardware completely secret-free as promising endeavor in future hardware designs, at least in those application scenarios where this is logically possible. In others, we suggest that secret-free techniques could be combined with standard PUFs and classical methods to construct hybrid systems with notably reduced attack surfaces.

Keywords Physical unclonable functions (PUFs) · Complex PUFs · SIMPL systems (SIMPLs) · Public PUFs (PPUFs) · Unique objects (UNOs) · Secret-free security

1 Introduction

1.1 Motivation and overview

At least since the introduction of Kerckhoffs' principle in the 19th century [35], the notion of cryptographic security has always been intimately tied to the concept of a secret key. Many classical cryptographic tasks were considered possible solely under the assumption that one party permanently stored a digital number that remained unknown to adversaries.

✉ Ulrich Rührmair
ruehrmair@ilo.de

[1] LMU München, Munich, Germany

[2] University of Connecticut, Storrs, USA

It is not too difficult to see that in a purely Turing-machine-based view of security, such keys are indeed *provably* indispensable. Take the task of secure identification as a simple example: If Alice is modeled as Turing machine, and if she securely wants to identify herself remotely to others, some "part" of her Turing program or Turing tape must be unknown to impersonators. This part automatically constitutes a *secret digital key* of the identification scheme, then. Similar considerations apply to other typical cryptographic tasks, such as encryption and decryption, message authentication, or digital signatures.

Unfortunately, this purely mathematical view comes along with some well-known implementation issues, because the effective safeguarding of secret digital keys in electronic hardware still constitutes a major challenge to this day. Both digital malware and various physical attacks have proven

highly efficient key extraction strategies over the years [1]. With the internet of things looming, the problem is exacerbated by the high level of mobility and interconnectivity of modern devices, as well as the complexity of their operating systems, programs, and concurrently running apps. This turns permanently stored, digital secret keys into one of the Achilles heels of contemporary security hardware.

The described situation has at least in parts motivated the introduction of physical unclonable functions (PUFs) over a decade ago [21,46]. Their central idea is to replace classical secret keys by complex physical structures, whose individual challenge–response behavior depends on unique and unclonable manufacturing variations in the PUF. In this fashion, PUFs can individualize and identify hardware without nonvolatile memory (NVM) or permanent digital keys on board [21,32,47], and can even be employed in various advanced cryptographic protocols, such as key exchange, oblivious transfer, or secure multi-party computation [6,61,68]. From the perspective of this paper, they could hence be seen as *avoiding* permanently stored digital secrets in vulnerable hardware.

At the same time, however, recent research has revealed that standard silicon PUFs cannot completely evade any sort of secrets in hardware yet: For example, the power-up states of SRAM PUFs [26,32], the individual runtime delays in Arbiter PUFs [21], or, generally speaking, the manufacturing variations in most silicon PUFs that determine their responses, still must remain *secret*, i.e., unknown to adversaries, for maintaining security. This observation has enabled a range of practically effective attacks on these PUFs [29,44,57,60,77], all of which have in common that they first identify, then extract various secrets from the PUFs, breaking their security.

A second and more recent class of physical security primitives, including so-called Complex PUFs, SIMPLs/ PPUFs [4,65], and UNOs, now shows promise to complete the endeavor originally started out by PUFs, however. Implemented successfully, they could maintain their security features under a surprisingly strong attack model: Even if the adversary was allowed to inspect their hardware bit by bit and atom by atom, and allowed to learn *any* information that is present in the hardware in *any* form at *any* time during a cryptographic scheme's execution, security would still be maintained. Such hardware therefore not just avoids permanently stored digital keys, but is completely *secret-free* for obvious reasons.

Being innately immune against any forms of key extraction, secret-free hardware could hence potentially lead to a disruptive change in the field. It would end the perpetual battle between key extractors and key protectors in a rather unexpected manner: Namely by simply removing any secrets, and thus any adversarial targets, from vulnerable systems, at least in certain application scenarios. This intriguing possibility, which is currently appearing almost unnoticedly on the horizon, motivates this survey and tutorial paper.

1.2 Related work

The best-known primitive by which "classical keys" (i.e., secret digital keys stored permanently in NVM) can be avoided, clearly are standard PUFs [21,40,47]; please also compare surveys [31,41,55,69] and references therein. Subsequent follow-up works on SIMPL Systems (SIMPLs) [65] and Public PUFs (PPUFs) [4] *implicitly* taught us how PUFs could even be made "secret-free" in our sense. The early publications on SIMPLs/ PPUFs did not make this aspect fully *explicit*, though, and demonstrably did not put it in their center; their focus lay on the public key functionality of the introduced SIMPLs/PPUFs instead [4,65].

It is an interesting historical footnote that already prior to PUFs, an almost unnoticed side strand of security research had suggested unclonable physical primitives that were not only free of any classical keys, but also free of any secrets, at least in the parlance of this paper. These early and sometimes rudimentary approaches usually focused on the forgery-proof labeling or tagging of valuable items, such as credit cards, passport, bank notes, paper stock certificates, branded consumer products, and the like, and thus arguably did not cover the full application spectrum of modern PUFs yet. They date back to as early as the late 1960s [39], and have resurfaced independently again in the 1980s [3,23], 1990s [10,27,71,74,83,86,87], and 2000s [7,9,11,14,28,70,89]. It seems that these methods remained more or less unnoted by the larger security community, though; to this day, they are not routinely quoted in the PUF-literature, unfortunately.

Another related research strand concerns the usage of general physical assumptions in *theoretical cryptography*. It, too, shares a long tradition, reaching back to the 1980s, and includes quantum [5], noise-based [43], DNA-based [12,37], or relativistic cryptography [34]. Its main motivation arguably lies in replacing the standard, unproven computational complexity assumptions of mathematical cryptography by other, independent physical hypotheses, however; its focus is not on the avoidance of classical keys or secrets in hardware. (The only exception known to us may be recent approaches on quantum read-out PUFs [25,73] and quantum public key encryption [82].)

Given this existing research landscape, it hopefully seems fair to say that none of the above publications explicitly makes the fundamental distinction between classical keys and secrets that underlies our paper, or solely focuses on the idea of secret-free hardware, as we do in this work. Concerning traceable historic roots, the first works in which the concept of secret-free security was fully and unambiguously expressed (sometimes without using exactly this terminol-

ogy) to our knowledge is [63,64] in 2011/2012.[1] The first formal definition of a secret-free primitive (in this case a secret-free so-called SIMPL system) apparently appeared in [63] in 2012.[2] Yet another, but later source known to us is [30] from 2016.

On a final note, this paper is an extended journal version of an earlier workshop publication [66] and a prior eprint [67] of the same author. Broadening the workshop publication, a new section on UNOs has been added (Sect. 6). Furthermore, this paper has been fundamentally rewritten in large parts, and various reworked or partly even new definitions have been added (e.g., Def. 3, 4, 8, 11, 14, 15, and Table 1).

1.3 Our contributions

We make the following contributions in this work:

- To the best of our knowledge, this is the first survey and tutorial paper on secret-free security.
- We propose semi-formal definitions of a "secret" and of "secret-free hardware."
- Furthermore, we develop the first formal taxonomy of different types of secrets.
- We use our taxonomy to classify the various types of secrets in several existing, exemplary PUF-schemes. This illustrates that standard PUF hardware is free of classical keys, but not free of secrets yet.
- We formally introduce the concept of Complex PUFs as a link between the world of standard PUFs and the secret-free world beyond, building on and extending an earlier analysis of Pappu et al. [46,47] on the simulation complexity of their optical PUF construction.
- We propose a general framework for defining secret-free primitives. It employs a game-based, parametric approach that considers finite, single physical objects (instead of infinite families of objects as in [6]). At its core, the framework employs the concept of a *complete internal state* of hardware: Providing the complete internal state of a hardware to attackers, while still requiring that security should be upheld, is suggested as general method for defining secret-free security.
- We then use our framework to formally define the secret-free primitives of Complex PUFs, SIMPLs/ PPUFs, and UNOs.
- We overview typical implementations, protocols, and applications of Complex PUFs, SIMPLs/ PPUFs, and UNOs.
- We also briefly survey other secret-free techniques, including virtual proofs of reality, secret-free sensing, digital rights management, or secret-free tamper detection.
- Finally, by putting secrets (instead of classical keys) into focus, we try to restructure the areas of PUFs and physical cryptography: We suggest that future approaches could try to *completely* avoid *any* secrets in hardware, at least wherever possible, instead of merely avoiding so-called "classical keys" (=digital keys stored permanently in non-volatile memory).

1.4 Organization of this paper

This manuscript is organized as follows: Sect. 2 provides a formal definition of secrets, a taxonomy of secrets, and definitions of secret-free hardware. Section 3 analyzes the various secrets in typical Weak and Strong PUF schemes. Sections 4, 5, and 6 detail three secret-free security primitives: Complex PUFs, SIMPLs/PPUFs, and UNOs. Section 7 describes possible extensions of secret-free security techniques beyond identification applications. Table 1 on page 23 provides a compact overview of our main results and classifications.

2 Defining secrets and secret-free security

2.1 Definition of secrets

This and the next two subsections will expound some of the central formal concepts of this work. As a preparatory step, it will be useful to stipulate the "standard" sequential security schemes considered throughout our paper.

Definition 1 (*Standard Sequential Security Schemes*). For the purposes of this paper, a so-called "standard" sequential security scheme $\mathcal{S}$ is assumed to possess the following properties:

- $\mathcal{S}$ is implemented by a finite number of hardware systems $\mathcal{H}_1, \ldots, \mathcal{H}_k$.
- $\mathcal{S}$ consists of an initial set-up phase, which is termed R_0 for consistency reasons, and a subsequent execution phase, which comprises a finite number of sequential runs $\mathsf{R}_1, \ldots, \mathsf{R}_n$.
- Each of these runs R_j (with $1 \leq j \leq n$) is expected to possess certain (general or individual) security features with respect to some adversary $\mathcal{A}_{\mathcal{S}}$. □

The above requirements are very mild, meaning that Definition 14 is hardly restrictive. Its purpose is mainly to provide some language for talking about secrets, as we do in the next definition.

[1] See abstract and page 2 of [64] and abstract and pages 2/3 of [63].

[2] See Specification 1 on pages 4/5 of [63].

Definition 2 (*Hardware Secrets*). Let $\mathcal{S}$ be a standard sequential security scheme. A finite number or bitstring sec is called a *hardware secret* (or just *secret*) with respect to $\mathcal{S}$ if:

- sec is physically represented in arbitrary form[3] in some hardware system $\mathcal{H}_i$ of $\mathcal{S}$ at least once during some run R_j of $\mathcal{S}$ (where R_j may also be the set-up phase R_0).
- The disclosure of sec to the adversary $\mathcal{A}_\mathcal{S}$ at the end of R_j allows $\mathcal{A}_\mathcal{S}$ to break some of the expected security features of at least one of the runs $\mathsf{R}_1, \ldots, \mathsf{R}_n$ (including R_j).

Under these circumstances, we will also call sec a secret in $\mathcal{H}_i$ (with respect to $\mathcal{S}$), or say that $\mathcal{H}_i$ contains the secret sec (during R_j). □

Slightly looking ahead, concrete examples of secrets will be provided to interested readers in Sects. 3, 4, or also Table 1. They include the physical manufacturing variations in SRAM PUFs or Arbiter PUFs; the power-up states of SRAM PUFs; the signal delays in Arbiter PUFs; the digital values of the sub-responses of all parallel single Arbiter PUFs in an XOR Arbiter PUF construction; the content of the arbiter elements (=latches) in XOR Arbiter PUFs; and, of course, any "classical" digital secret keys that are stored permanently in hardware. Readers familiar with PUFs may perhaps find these examples useful to build some first intuition.

But for now, let us remain on an abstract level first, and discuss on several formal aspects and peculiarities of Definition 2. Firstly, the concept of a *secret* obviously only makes sense in relation to a given security scheme $\mathcal{S}$ and its expected security properties. No piece of information is a secret just by itself, but only becomes one in a given context. The definition takes this into account, defining secrets sec relative to schemes $\mathcal{S}$.

Secondly, in order to achieve full generality, Definition 2 must leave the realm of the Turing formalism: As PUFs and other physical security primitives have taught us, secrets need not always be classical keys, i.e., secret digital numbers stored permanently in memory. This prevents the standard application of the Turing machine in Definition 2, while leaving open which other formalism should be employed instead. The utilization of *physical* Turing machines, as carried out in [62], would be exact and rigorous, but also cumbersome, creating involved language and expressions. The best actual remedy seems the precisest possible use of everyday language, as employed in the definition. This holds in particular with hindsight to the question what it means that some information is "physically represented" in a hardware system (please reconsider Footnote 3 in this context). We comment that this approach does have some tradition in theoretical computer science; compare, for example, the Church–Turing thesis [90], which must necessarily be formulated in everyday language. While there are alternative options, our approach has the advantage of creating a rigorous, yet easily accessible framework.

Thirdly, it may seem surprising that the definition uses a mild form of time line, stipulating that adversaries are only given a secret sec *after* the execution of a certain run R_j, not *during* it. The reason is relatively straightforward: Consider for illustration a standard, challenge–response type symmetric identification scheme [1], where verifier and prover hold the same key K. In each run, the verifier will send a fresh, random nonce N to the prover, whose hardware computes $\mathsf{PRF}_\mathsf{K}(\mathsf{N})$ for a keyed pseudo-random function PRF, and returns this value to the verifier. Now, if the value $\mathsf{PRF}_\mathsf{K}(\mathsf{N})$ (which must be computed by and thus contained in the prover's hardware during each run), was given to an adversary *immediately* during this very run, impersonation would become straightforward: The adversary could simply instantly pass on $\mathsf{PRF}_\mathsf{K}(\mathsf{N})$ to the verifier, impersonating the prover in this protocol run.

A definitional framework that would allow adversaries to learn any potential secrets *instantaneously* during a run R_j, i.e., as soon as they appear in the prover's hardware, would therefore formally turn the value $\mathsf{PRF}_\mathsf{K}(\mathsf{N})$ into a "secret" of said identification scheme. While intuitively, and in any natural understanding of the term "secret," it should not be considered as such: Recall that $\mathsf{PRF}_\mathsf{K}(\mathsf{N})$ is sent in the clear over the communication channel from the prover to the verifier in the course of the identification protocol.

Definition 2 prevents this paradox by stipulating that the attacker shall only learn potential secrets *after* the end of the respective run R_j. In effect, this ensures that a secret is something deeper than just a (public) message in a challenge–response protocol that is passed on or replayed. It instead must be a value that has a more profound impact, either backward on the security of already completed runs, or on the future security of yet unstarted runs (in which new randomness or new challenges are used).

Finally, formulating our definition relative to the expected security properties and the adversary $\mathcal{A}_\mathcal{S}$ of $\mathcal{S}$ makes the definition more general, allowing broad applicability to not yet fully specified future schemes. This is one other, and final, motivation for defining secrets relative to schemes $\mathcal{S}$.

2.2 A taxonomy of secrets

The recently emerging physical security techniques, including PUFs, have taught us that there are different "classes"

[3] This naturally includes, but is not limited to, the content of volatile or nonvolatile memory, any physical properties of circuit components, any physical characteristics of transient signals (including signal timing, shape or strength), as well as any internal physical states, structures, or configurations of the hardware.

or "types" of secrets in hardware. This motivates the development of a *taxonomy of secrets* below. The taxonomy will later be applied to distinguish various types of secrets in the upcoming sections.

Definition 3 (*A Taxonomy of Secrets*). Let sec be a secret in some hardware system $\mathcal{H}_i$ with respect to some standard sequential security scheme $\mathcal{S}$. Then, we call sec a:

(i) **Permanent digital secret** if it is represented permanently in digital form (i.e., in nonvolatile or volatile digital memory) in $\mathcal{H}_i$ during all runs $\mathsf{R}_1, \ldots, \mathsf{R}_n$.

(i.a) *Permanent, nonvolatile digital secret*, or simply a *classical key*, if it is represented permanently in nonvolatile digital memory in $\mathcal{H}_i$ during all runs $\mathsf{R}_1, \ldots, \mathsf{R}_n$.

(i.b) *Permanent, volatile digital secret* if it is represented permanently in volatile digital memory in $\mathcal{H}_i$ during all runs $\mathsf{R}_1, \ldots, \mathsf{R}_n$.

(ii) **Non-permanent digital secret** if it is represented at least once, but not permanently, in digital form (i.e., in nonvolatile or volatile digital memory, or in the digital values of transient digital signals) in $\mathcal{H}_i$ during at least one run $\mathsf{R}_0, \ldots, \mathsf{R}_n$.

(ii.a) *Non-permanent, nonvolatile digital secret* if it is represented at least once, but not permanently, in nonvolatile digital memory in $\mathcal{H}_i$ during at least one run $\mathsf{R}_0, \ldots, \mathsf{R}_n$.

(ii.b) *Non-permanent, volatile digital secret* if it is represented at least once, but not permanently, in volatile digital memory in $\mathcal{H}_i$ during at least one run $\mathsf{R}_0, \ldots, \mathsf{R}_n$.

(ii.c) *Transient digital secret* if it is represented at least once, but not permanently, in the digital values of transient digital signals in $\mathcal{H}_i$ during at least one run $\mathsf{R}_0, \ldots, \mathsf{R}_n$.

(iii) **Digital secret** if it is any of the above types (i) to (ii.c), and **physical secret** if it is not a digital secret.

(vi) **Permanent physical secret** if it is a physical secret that is represented in $\mathcal{H}_i$ permanently during all runs $\mathsf{R}_1, \ldots, \mathsf{R}_n$.

(vi.a) *Permanent, nonvolatile physical secret* if it is a physical secret that is represented in $\mathcal{H}_i$ permanently during all runs $\mathsf{R}_1, \ldots, \mathsf{R}_n$, and in such a way that it remains present if all power and energy supplies for $\mathcal{H}_i$ are disabled.

(vi.b) *Permanent, volatile physical secret* if it is a physical secret that is represented in $\mathcal{H}_i$ permanently during all runs $\mathsf{R}_1, \ldots, \mathsf{R}_n$, and in such a way that it is lost if all power and energy supplies for $\mathcal{H}_i$ are disabled.

(v) **Non-permanent physical secret** if it is a physical secret that is represented at least once, but not permanently, in $\mathcal{H}_i$ during at least one run $\mathsf{R}_0, \ldots, \mathsf{R}_n$.

(v.a) *Non-permanent, nonvolatile physical secret* if it is a physical secret that is represented at least once, but not permanently, in $\mathcal{H}_i$ during at least one run $\mathsf{R}_0, \ldots, \mathsf{R}_n$, and in such a way that it remains present if all power and energy supplies for $\mathcal{H}_i$ are disabled.

(v.b) *Non-permanent, volatile physical secret* if it is a physical secret that is represented at least once, but not permanently, in $\mathcal{H}_i$ during at least one run $\mathsf{R}_0, \ldots, \mathsf{R}_n$, and in such a way that it is lost if all power and energy supplies for $\mathcal{H}_i$ are disabled. □

Sections 3, 4, and Table 1 will animate Definition 3: They will present concrete examples for all mentioned types of secrets. But for now, let us better remain on an abstract level for a while, discussing the definition's formal approach in all detail.

The definition distinguishes secrets along the following three dimensions: *(a)* Are they digital or physical? *(b)* Are they permanent or non-permanent? *(c)* Are they nonvolatile, volatile, or (in the case of digital signals) transient? This detailed differentiation leads to an implicit "hierarchy" of secrets, which ranges from the generally more vulnerable ones (item (i.a)) to the commonly more attack-resilient ones (item (v.b)).

Let us expound this idea of a general "hierarchy" among secrets in a bit more detail. To start with, it seems reasonably to claim that *digital* secrets (as defined in items (i) to (iii) of Definition 3) at least generally are more vulnerable than *physical* ones: Information present in digital form typically appears easier to extract than arbitrary analog/physical characteristics [1,78]. Secondly, *permanent* secrets seem, at least in principle and in general, more vulnerable than *non-permanent* ones: The obvious reason is that they are present in hardware for (sometimes substantially) longer periods.

Thirdly, *nonvolatile* secrets typically appear more vulnerable than their *volatile* counterparts, which, in turn, commonly are better accessible than secrets that exist only in the form of *transient* signals. One reason is that nonvolatile secrets remain in the system if the power supply is turned off, cut, or destroyed during an attack, or if the attacked hardware is (partly) disassembled [1,78]. Nonvolatile memory also commonly shows stronger data remanence effects than volatile memory [1,78]. One further reason is that extracting transient digital signals on the fly generally seems more cumbersome for adversaries than reading out static digital memory [1,78].

All these arguments seem to add to the idea of a potential hierarchy of secrets in Definition 3, ranging from the generally more vulnerable secrets (such as (i.a)), to the more attack resilient ones (such as (v.b)). While this hierarchy is by no

means fully strict or binding, and is necessarily subject to all kinds of exceptions and counterexamples[4], it still can provide a useful guideline for a first assessment of the resilience of given hardware against physical attacks. Using our taxonomy, an analysis of the various secrets in new architectures could be standardized easily, pointing to possible vulnerabilities at early design stages. Interestingly, the actual historic development in cryptography and hardware security has relatively closely followed our mainly theoretically inspired taxonomy (please compare Table 1 on page 23). This could potentially be seen as another reason supporting our classification and hierarchy.

Let us next dive yet deeper into some other details of Definition 3, which are independent of the above idea of a potential hierarchy. Firstly, readers will hopefully agree that the concept of a transient signal is meaningful only if the hardware has some circuit-like structure, i.e., if it has a *digital* layer. Furthermore, it only makes sense in connection with *non-permanent* secrets. For these reasons, transient digital signals are only defined as a sub-concept of non-permanent digital secrets, and not in other contexts, in item (ii.c) of Definition 3.

Secondly, readers may have noticed that all permanent secrets are defined over the runs $\mathsf{R}_1, \ldots, \mathsf{R}_n$ (i.e., *excluding* the set-up phase R_0), while all non-permanent secrets are defined over all runs $\mathsf{R}_0, \ldots, \mathsf{R}_n$ (i.e., *including* the set-up phase R_0). The reason is that even permanent secrets need to be generated at some point during a security scheme, and this usually takes place within the set-up phase. Requiring that permanent secrets should already be present during the entire set-up phase, i.e., form its very start onward, would simply ask for too much. On the other hand, *non-permanent* secrets may be present temporarily in the set-up phase R_0 (and perhaps not even in any other runs of the scheme![5]). This suggests to explicitly allow this possibility in our definitional framework, including the set-up phase R_0 in our definition for the case of non-permanent secrets.

Thirdly, Definition 3 makes every effort to meaningfully transfer the concept of nonvolatile/volatile *digital* secrets to the more general situation of *physical* secrets. We stipulate that in physical hardware layers, nonvolatile secrets must remain present once any energy supply is disabled, while volatile secrets are lost in this case. The "energy supply" in our sense need not be a standard electrical signal, but could also be a laser, a dedicated thermal energy source, etc. We hope that this keeps our definition general and applicable in different circumstances.

2.3 Key-free and secret-free hardware and their promise

As already mentioned, Definition 3 pinpoints various types of secrets, introducing some mild hierarchy among them. This suggests a complementary hierarchy of hardware systems that are *free* of the respective secrets: For example, hardware free of permanent digital secrets; free of permanent, volatile digital secrets; free of permanent physical secrets; etc. Two particular elements of this "mirrored" and complementary hierarchy possess special importance and relevance for us, namely "key-free" and "secret free" hardware systems. Formally defining and comparing them will be the topic of this brief section.

Definition 4 (*Key-Free Hardware*) A single hardware system $\mathcal{H}_i$ of a standard security scheme $\mathcal{S}$ is called *free of classical keys* (or simply *key-free*) if it contains no classical keys (with respect to $\mathcal{S}$). □

Along the same lines, secret-free hardware can be defined as follows:

Definition 5 (*Secret-Free Hardware*) A single hardware system $\mathcal{H}_i$ of a standard security scheme $\mathcal{S}$ is called *secret-free (with respect to $\mathcal{S}$)* if it contains no secrets (with respect to $\mathcal{S}$). □

The above two definitions naturally imply that any secret-free hardware is also key-free. The converse is not true, as Sects. 3 and 4 show in all detail: Many hardware systems based on standard PUFs are key-free, but not completely secret-free yet.

Let us again lead an abstract, comparative discussion between key-free and secret-free hardware below, carefully illustrating their mutual differences and respective advantages. This will help us to gain momentum for the rest of the paper, and will motivate the notable benefits of secret-free over merely key-free hardware.[6]

To start with, key-free hardware surely promises less vulnerabilities against physical attacks than systems containing classical keys. The simple reason is that attackers must extract other types of secrets, such as non-permanent, volatile, or physical ones. These are usually either located on "deeper," physical hardware levels, and/or are present in the system only for shorter periods of time. This constitutes a notable achievement of key-free systems, and could be regarded as the initial PUF-promise.

[4] It is not too difficult, for example, to imagine poorly designed physical secrets, which are far simpler to extract than a well-protected classical key, and so on.

[5] See Sect. 4.4 and Scheme 10 for an example: The laser beam that constitutes the challenges $C_1, \ldots, C_{\lambda \cdot n}$, which are applied to the scattering token/optical PUF during the set-up phase R_0, by Definition 2 is only a non-permanent, volatile physical secret in R_0 (and not in the other runs $\mathsf{R}_1, \ldots, \mathsf{R}_n$ of the scheme)

[6] Once more, concrete examples will be provided throughout the following Sects. 3 to 7.

At the same time, latest PUF-research has revealed that the above-mentioned attacks on deeper system levels or on non-permanent secrets may be hard, but are not impossible [29,44,57,60,77]. Key-free hardware systems which still contain secrets may therefore improve attack resilience, constituting a strong and notable achievement of such systems and of standard PUFs in general. But still, they often cannot evade physical attacks completely yet.

Secret-free systems try to take this approach one critical step further: They promise that even adversaries who could hypothetically access *any* information that is present in hardware in any, possibly physical form during a security scheme (not just in the form of digital, permanently stored keys in nonvolatile memory!), and who could unrestrictedly inspect *every* bit and even *every* atom of the system, would be unable to break security. Secret-free hardware thus possesses provable and complete immunity against any type of adversarial probing, and against software-based key-extraction as well, including malware—the obvious reason being that there simply is nothing security–critical to extract.

Furthermore, secret-free security systems may not even require confidentiality in their set-up phases (compare Scheme 13 as example). The plain reason is that there are no secrets in the set-up phases that need to be protected. They thus can establish certain partial security guarantees even against malicious manufacturers. Their setup could, for example, be accomplished under the eyes of the public, or in the presence of cryptographic adversaries, should this become necessary. This feature can enable new classes of security schemes: For example, in the context of nuclear weapons inspections, set-up phases for the inspectors' equipment may need to be conducted in the potentially hostile environments of the inspected nation [48].

Taken together, this makes secret-free security a worthwhile theoretical concept to pursue. To which extent it can already be realized in practice, and in which scenarios it is already concretely applicable (and in which not), will hopefully become clearer throughout our detailed practical discussions in the upcoming Sects. 3 to 7.

3 Secrets in standard PUFs

Having laid the theoretical foundations in Sect. 2, we will now shift our focus, moving to more practical topics. To start with, this Sect. 3 will illustrate our taxonomy of secrets at work, and present concrete examples for various types of secrets. More specifically, we will identify miscellaneous secrets in protocols built on the two most widespread silicon PUFs: Namely in identification protocols built on SRAM PUFs [26,32] and on XOR Arbiter PUFs [21,76]. This will again confirm our earlier mantra: While standard PUFs are free of classical keys, they are not secret-free yet.

3.1 Basics of SRAM PUFs and XOR Arbiter PUFs

To make this paper as self-contained as possible, we will start our discussion with a rudimentary introduction to the above-mentioned two PUF types; interested readers are referred to existing PUF-surveys [31,41,55,69] for further details.

In a nutshell, an SRAM PUF is nothing else than a standard array of k SRAM cells. Each single cell possesses a specific and (hopefully) reasonably stable power-up behavior, which is determined by its random manufacturing variations. The so-called "PUF-challenge" in the case of an SRAM PUF array is simply this power-up operation, i.e., the entire PUF merely has one challenge. The "PUF-response" is the concatenated string of the k power-up values of the k SRAM cells. SRAM PUFs belong to a PUF-class that has been termed Weak PUFs [55] or Physically Obfuscated Keys (POKs) [31] at times.

An XOR Arbiter PUF, on the other hand, is a construction that consists of k parallel, single Arbiter PUFs [76]. It possesses exponentially many possible challenges C_i, more precisely 2^n challenges, where n is its challenge length; usually $n = 64$ or larger. A given challenge C_i is applied equally and identically to all these k parallel Arbiter PUFs, producing k sub-responses $R_i^1, \ldots, R_i^k$. These sub-responses are held temporarily in the nonvolatile arbiter elements/latches at the end of each Arbiter PUF. The global, overall PUF-response R_i of an XOR Arbiter PUF is then digitally computed on chip as the XOR of these k sub-responses $R_i^1, \ldots, R_i^k$. Physically, this is accomplished by a large XOR-gate connected to all the latches, i.e., $R_i = R_i^1 \oplus \cdots \oplus R_i^k$.

XOR Arbiter PUFs have originally been suggested as candidates for a PUF-class known as Strong PUFs [31,55]. Interested readers are again referred to existing PUF-surveys for further details [31,41,55,69].

Definition 2 tells us that for a well-founded analysis of secrets in PUF-hardware, we also need to pick a specific security scheme $\mathcal{S}$ in which this hardware is used. As a natural choice here (and in the largest part of this paper), we selected the remote identification of a prover to a verifier over a digital channel [1]. The simple reason is that this protocol constitutes the historically first and still most popular PUF application to this day [21,47], and that it is conceptually relatively easy to describe.

3.2 Secrets in SRAM PUF-based identification

Let us then start by analyzing the secrets induced by SRAM PUFs [26,32] in remote identification. The standard scheme to this end [26,32] assumes that an SRAM PUF (consisting of k SRAM cells) is contained and hardwired in the prover's

hardware $\mathcal{H}_\mathsf{P}$.[7] The power-up states of these k cells are used to derive a key K. The prover's hardware $\mathcal{H}_\mathsf{P}$ subsequently uses this key in a classical symmetric identification protocol with the verifier's hardware $\mathcal{H}_\mathsf{V}$.

For the sake of an accurate analysis, and again in order to make this paper self-contained, let us spell out the various details of the scheme as follows:

Scheme 6 (Identification with SRAM PUFs [26,32])

Set-Up Phase (also called R_0):

1. The prover's hardware $\mathcal{H}_\mathsf{P}$ internally measures the k (noisy) digital power-up states of the SRAM PUF.
2. $\mathcal{H}_\mathsf{P}$ provides these power-up states to the verifier's hardware $\mathcal{H}_\mathsf{V}$ via some digital communication interface. $\mathcal{H}_\mathsf{P}$ then erases these power-up states internally (i.e., in its own system $\mathcal{H}_\mathsf{P}$), for example, by overwriting them with zeros.
3. $\mathcal{H}_\mathsf{V}$ derives a stable secret key K and error-correcting helper data HD from said power-up states. $\mathcal{H}_\mathsf{V}$ then stores the resulting key K permanently in nonvolatile digital memory.
4. Next, $\mathcal{H}_\mathsf{V}$ provides the helper data HD to the prover's hardware $\mathcal{H}_\mathsf{P}$ via the above-mentioned digital communication interface. $\mathcal{H}_\mathsf{P}$ stores HD permanently in nonvolatile digital memory.
5. Finally, the functionality of $\mathcal{H}_\mathsf{P}$ by which the power-up states of the k SRAM cells may be provided to external parties like $\mathcal{H}_\mathsf{V}$ during the set-up phase is irreversibly disabled.

Execution Phase (Run R_i, with $1 \leq i \leq n$):

1. The verifier's hardware $\mathcal{H}_\mathsf{V}$ remotely sends a fresh random nonce $\mathsf{N_i}$ to the prover's hardware $\mathcal{H}_\mathsf{P}$.
2. $\mathcal{H}_\mathsf{P}$ triggers a power-up operation of its SRAM cells. It derives the key K from the power-up states of the k SRAM cells of its SRAM PUF. In this context, $\mathcal{H}_\mathsf{P}$ uses the stored helper data HD and some error-correction mechanisms. Finally, $\mathcal{H}_\mathsf{P}$ stores K in volatile memory.
3. $\mathcal{H}_\mathsf{P}$ computes the value $\mathsf{PRF_K(N_i)}$ for some pseudorandom function PRF. To this end, the key K is transferred from the volatile memory in which it was stored to the part of the hardware that computes PRF.
4. Next, $\mathcal{H}_\mathsf{P}$ sends the value $\mathsf{PRF_K(N_i)}$ remotely to the verifier's hardware. After this, $\mathcal{H}_\mathsf{P}$ erases the key K and the power-up states of the SRAM cells, e.g., by overwriting them with zeros.
5. The verifier's hardware checks the correctness of the received function value $\mathsf{PRF_K(N_i)}$, using K. It accepts the identification if and only if the value is correct. □

It is clear that the prover's hardware $\mathcal{H}_\mathsf{P}$ is free of any classical keys, i.e., of any permanent nonvolatile digital secrets, while the verifier's hardware $\mathcal{H}_\mathsf{V}$ does contain such classical keys. Applying the taxonomy of Definition 3 enables a yet far more finegrained analysis, however. It allows us to identify various *other* types of secrets in $\mathcal{H}_\mathsf{P}$ and $\mathcal{H}_\mathsf{V}$. Their disclosure to adversaries would break the scheme as well, and would allow impersonation of the prover in future protocol runs.

Let us start with the secrets in *the prover's hardware* $\mathcal{H}_\mathsf{P}$:

- $\mathcal{H}_\mathsf{P}$ is **free of classical keys** (or of permanent nonvolatile digital secrets in our parlance).
- Still, $\mathcal{H}_\mathsf{P}$ stores the key K in volatile memory for some limited time in every run R_i (for $1 \leq i \leq n$). This makes K a **non-permanent, volatile digital secret** in our terminology.
- $\mathcal{H}_\mathsf{P}$ contains another **non-permanent, volatile digital secret** in every run R_i (this time even *including* the set-up phase, i.e., for $0 \leq i \leq n$), namely the power-up states of the k SRAM cells.
- In the case that $\mathcal{H}_\mathsf{P}$ is a *completely digital system*, it will furthermore contain various **transient digital secrets**:
  As first example, in the set-up phase R_0, the k power-up states will be transferred via the digital interface from $\mathcal{H}_\mathsf{P}$ to $\mathcal{H}_\mathsf{V}$. These power-up states hence also constitute transient digital secrets in $\mathcal{H}_\mathsf{P}$.
  Secondly, in each run R_i (with $1 \leq i \leq n$), the key K will be transferred by transient digital signals within $\mathcal{H}_\mathsf{P}$, namely from the volatile memory where it is stored to the circuitry that computes $\mathsf{PRF_K(N_i)}$ (Step 3 of the execution phase of Scheme 6). This makes K a transient digital secret, too.
  As third example, in each run R_i (with $0 \leq i \leq n$), the digital signals that communicate the power-up states of the SRAM cells to the error-correction/key derivation mechanism inside $\mathcal{H}_\mathsf{P}$ constitute transient digital secret as well.
- Finally, $\mathcal{H}_\mathsf{P}$ contains **permanent, nonvolatile physical secrets**: Namely the physical manufacturing variations that determine the power-up states of the SRAM cells. Knowing them, adversaries could derive the power-up states of the SRAM cells by numerical simulation in practice. They may then infer K themselves (recall that the helper data HD and the method for error-correction usually are assumed to be public), breaking security.

Let us continue with the secrets in the *verifier's hardware* $\mathcal{H}_\mathsf{V}$:

[7] Just to quickly remind the reader again: The party who wants to identify itself in a remote identification protocol is usually called the "prover" in cryptographic parlance. The party who wants to verify the identify of the prover is termed the "verifier."

– The *verifier's hardware* $\mathcal{H}_\mathsf{V}$ obviously contains a **classical key** (or, in the language of Definition 3, a permanent, nonvolatile digital secret), namely the secret key K.
– If $\mathcal{H}_\mathsf{V}$ is a *fully digital system*, it will contain various **transient digital secrets**, too: Just to name a first example, in run R_0, transient digital signals will deliver the power-up states of the k SRAM cells key K inside $\mathcal{H}_\mathsf{V}$ from the communication interface with $\mathcal{H}_\mathsf{P}$ to the nonvolatile memory of $\mathcal{H}_\mathsf{V}$, where it is stored permanently (Step 3 of Scheme 6).
As a second example, in each single run R_i (for $1 \leq i \leq n$), the key K is digitally transferred from its permanent storage location in $\mathcal{H}_\mathsf{V}$ to the module inside $\mathcal{H}_\mathsf{V}$ where the prover's answer $\mathsf{PRF}(\mathsf{N_i})$ is verified. This again constitutes a digital transient secret in $\mathcal{H}_\mathsf{V}$ in each of these single runs.

The above analysis illustrates the potential of the new formalism: While from the perspective of classical keys, all to be said about Scheme 6 is that $\mathcal{H}_\mathsf{P}$ does not contain such classical keys, while $\mathcal{H}_\mathsf{V}$ in fact does, a secret-based analysis directly pinpoints deeper attack vectors and vulnerabilities. In fact, some of these vectors have been exploited in practical attacks in the past already: For example, the power-up states of SRAM PUFs have been read out invasively via smart techniques [29,44]. While these attack vectors were also known prior our taxonomy, Definitions 2 and 3 still seem to make a systematic and standardized analysis much easier.

On a short side note, the situation changes slightly, but not fundamentally if SRAM PUFs are used to derive a hardware-internal *(public key, private key)*-pair [81], which is subsequently used in an asymmetric identification scheme. In this case, the verifier's hardware becomes secret-free, as it merely needs to store a public key—but all above-mentioned secrets in the prover's hardware, which is usually more exposed and attack-prone in practical applications, remain unchanged.

3.3 Secrets in XOR Arbiter PUF-based identification

Will XOR Arbiter PUFs fare better concerning the types of secrets they cause in hardware? After all, their large number of challenge–response pairs (CRPs), together with their (supposedly) unpredictable responses [55], allow a different type of identification protocol [47].

In the upcoming Scheme 7, the XOR Arbiter PUF itself is employed as if it was a unique and unclonable pseudo-random function, which is contained/hardwired in the prover's hardware $\mathcal{H}_\mathsf{P}$.[8] Its CRPs are sent in the clear over the digital communication channel, without deriving internal secret keys first, as in Scheme 6. This approach is typical for the class of so-called Strong PUFs [55], for which XOR Arbiter PUFs were initially designed, and could be called "plain CRP-based identification" [47,55].

Scheme 7 provides all the further details. It employs a security parameter λ, and has an envisaged number of n runs.

Scheme 7 (Identification with XOR Arbiter PUFs [47])

Set-Up Phase (also called R_0):

1. The verifier's hardware $\mathcal{H}_\mathsf{V}$ chooses $\lambda \cdot n$ random challenges $C_1, \ldots, C_{\lambda \cdot n}$.
2. $\mathcal{H}_\mathsf{V}$ applies these challenges $C_1, \ldots, C_{\lambda \cdot n}$ to the XOR Arbiter PUF via its digital challenge–response interface. The corresponding responses $R_1, \ldots, R_{\lambda \cdot n}$ are collected via the same interface.
3. $\mathcal{H}_\mathsf{V}$ permanently stores the so created CRP-List $(C_1, R_1), \ldots, (C_{\lambda \cdot n}, R_{\lambda \cdot n})$ in nonvolatile memory.

Execution Phase (Run R_i, with $1 \leq i \leq n$):

1. The verifier's hardware $\mathcal{H}_\mathsf{V}$ randomly selects λ new CRPs $(C_1^i, R_1^i), \ldots, (C_\lambda^i, R_\lambda^i)$ from the CRP-List.
2. $\mathcal{H}_\mathsf{V}$ sends $C_1^i, \ldots, C_\lambda^i$ to the prover's hardware $\mathcal{H}_\mathsf{P}$.
3. $\mathcal{H}_\mathsf{P}$ applies these challenges to the XOR Arbiter PUF, and sends the obtained responses $\bar{R}_1^i, \ldots, \bar{R}_\lambda^i$ to $\mathcal{H}_\mathsf{V}$.
4. $\mathcal{H}_\mathsf{V}$ checks if the received responses $\bar{R}_1^i, \ldots, \bar{R}_\lambda^i$ match the pre-recorded responses $R_1^i, \ldots, R_\lambda^i$ from the CRP-List. It accepts the identification if and only if they match within a certain, previously specified error margin [47].
5. The CRPs $(C_1^i, R_1^i), \ldots, (C_\lambda^i, R_\lambda^i)$ are removed from the CRP-List. □

Scheme 7 induces various *secrets* in the hardware of the involved parties. Let us again start with the *prover's hardware* $\mathcal{H}_\mathsf{P}$:

– $\mathcal{H}_\mathsf{P}$ is **free of classical keys**, i.e., of permanent nonvolatile digital secrets.
– If the number n of sequential runs in Scheme 7 is large enough, $\mathcal{H}_\mathsf{P}$ will contain **non-permanent volatile digital secrets**, however: Namely the digital content of the latches which act as "arbiter elements" at the end of each single Arbiter PUF within the larger XOR Arbiter PUF construction.
Once many of these values are known to adversaries, machine learning of the single Arbiter PUFs, and subsequent prediction of the complete XOR Arbiter PUF, becomes possible [57]. For the same reason, the digital signals that enter the final XOR gate of the XOR Arbiter

[8] To again quickly remind the reader: The party who wants to identify itself in a remote identification protocol is usually called the "prover" in cryptographic parlance. The party who wants to verify the identify of the prover is termed the "verifier."

PUF are transient digital secrets. Interestingly, the secret in the two latter cases is not contained in a single run in $\mathcal{H}_\mathsf{P}$, but is spread over many runs R_i of Scheme 7.
- Closely following Definitions 2 and 3, $\mathcal{H}_\mathsf{P}$ also contains a **transient digital secret** in its set-up phase R_0, namely the CRPs $(C_1, R_1), \ldots, (C_{\lambda \cdot n}, R_{\lambda \cdot n})$ that are collected via the digital challenge–response interface of the XOR Arbiter PUF to create the CRP-List. Knowing these CRPs after the setup R_0, but before the runs R_j, allows impersonation of the prover's hardware $\mathcal{H}_\mathsf{P}$ in Scheme 7.
- Finally, $\mathcal{H}_\mathsf{P}$ contains **permanent nonvolatile physical secrets**: The manufacturing variations that cause the runtime delays in the XOR Arbiter PUF.

Let us continue with the secrets in the *verifier's hardware* $\mathcal{H}_\mathsf{V}$:

- $\mathcal{H}_\mathsf{V}$ contains a **permanent nonvolatile digital secret**, namely the CRP-List.
- $\mathcal{H}_\mathsf{V}$ does not contain any volatile or transient digital secrets, though: Scheme 7 only needs to read those CRPs from the CRP-List that are used in the current single run R_i. If these are revealed to the adversary *at the end* of R_i, as stipulated in Definition 2, this does not affect the security of R_i, and neither of any past or future runs. Recall that any CRPs employed in R_i are removed from the CRP-List, and are never re-used in future runs (compare Step 5 of the execution phase of Scheme 7 and Definition 2).

In other words: If Strong PUFs are implemented by XOR Arbiter PUFs and comparable designs [8,36,88], and used in plain CRP-based identification, they can avoid permanent nonvolatile digital secrets in the prover's hardware. But they are unable yet to realize secret-free security for the verifier or the prover.

3.4 Practical relevance of secrets and enduring benefits of standard PUFs

Let us conclude this section by discussing the practical relevance of secrets in PUF-based hardware and PUF-based protocols. It could be tempting to understand the distinction between classical keys and secrets as purely academic topics, claiming that it had no "real" relevance for practical hardware security. We would like to emphasize that this is not the case. Most of the above-mentioned secrets have actually been extracted from real silicon SRAM PUFs and XOR Arbiter PUFs, a fact that has rendered both PUF-types vulnerable or even insecure in practice [29,44,57,60,77]. In various cases, demarcating merely key-free from completely secret-free systems has been equal to drawing the line between insecure and secure systems. This observation holds not just in theory, but in practice [29,44,57,60,77].

Still, in order to obtain a balanced view, it seems important to again stress the enduring achievements of standard PUFs. They have been the first physical security primitive which enabled the individualization of hardware without nonvolatile memory on board, and which enabled security without classical keys. In the language of Definitions 3 and 4, they were hence the first primitives that could realize **key-free security** in certain applications and certain pieces of hardware. This constitutes a strong achievement in itself.

Diving further into the details, Strong PUF at least conceptually holds a yet stronger promise: Not just key-free security, but also security without any digital secrets in the Strong PUF carrying hardware (compare again Definition 3). Most existing silicon Strong PUF designs do not realize this property yet, as they possess digital sub-signals that must remain secret (such as the sub-responses in an XOR Arbiter PUF that enter the XOR gate, for example). But future, improved electrical Strong PUF designs might. Finally, optical Strong PUF constructions, such as Pappu et al.'s design [47], can already now realize certain forms of secret-free security, provided that they fulfill the additional criterion of being a Complex PUFs (see Sect. 4). They hence clearly foreshadow the later idea of secret-free security.

These benefits of standard PUFs are highly noteworthy, and will always remain, no matter which future development the area might take. We considered it important to stress this at the end of this section.

4 Secret-free security by complex PUFs

4.1 Definition of complex PUFs

Trying to leave the realm of key-free hardware systems, the perhaps easiest route to secret-free security is the concept of a *Complex PUF*, which we will formally introduce in this section. Complex PUFs are a special sub-class of so-called Strong PUFs [55], i.e., by definition they must possess a large and practically inexhaustible CRP-space, a publicly accessible challenge–response interface/mechanism, and responses that are difficult to predict numerically, even if many other challenge-response pairs are known (compare [55]).

On top of these features, a so-called $(\epsilon, t_{\mathsf{Pred}})$-*Complex PUF* shall fulfill the following property: Even if the adversary knows its *complete internal state* (including any manufacturing variations and internal random structure and parameters, such as resistances, threshold voltages, or signal delays) of an $(\epsilon, t_{\mathsf{Pred}})$-Complex PUF, and has some feasible time for preparatory calculations, he *cannot* numerically predict the correct response R_i to a randomly chosen challenge C_i with a probability *better than* ϵ and *within time* t after C_i has been presented to him.

This basic idea is more formally developed over the following two definitions.

Definition 8 (*Complete Internal State of a Hardware*). Let H be some hardware. The *complete internal state of* H, termed $\mathsf{CIS(H)}$, is a bitstring that describes all information that is present in H, and that is relevant for the functionality and security of H. This explicitly includes, but is not limited to:

- The digital design and system-level architecture of H, if it has any.
- The digital memory content of H, if there is any.
- The physical structure and physical state of H, including any manufacturing variations that are relevant for the functionality and security of H. □

The above concept of a complete internal state will be central for defining secret-free security. Similarly to Definition 2, it touches upon very fundamental questions—and is hard to formalize in compact form and with full rigor at the same time. As earlier, we hope that Definition 8 strikes a good balance between exactness and simple accessibility, using everyday language in precise manner (please also compare our discussion after Definition 2).

Definition 8 can be applied in the context of Complex PUFs as follows:

Definition 9 ($(\epsilon, t_{\mathsf{Pred}})$-*Secure Complex PUFs*). Let P be a PUF with complete internal state $\mathsf{CIS(P)}$. P is called an $(\epsilon, t_{\mathsf{Pred}})$-*secure Complex PUF (or just* $(\epsilon, t_{\mathsf{Pred}})$-*Complex PUF) with respect to an adversary* $\mathcal{A}$ if $\mathcal{A}$ has a probility of at most ϵ to win the following game:

FastPredGame(P, $\mathcal{A}$, $\mathsf{CIS(P)}$, t_{Pred}):

Phase 1: Preparation. $\mathcal{A}$ is given $\mathsf{CIS(P)}$ and physical access to P for one year. Throughout this period, $\mathcal{A}$ may conduct physical measurements on P (including determination of CRPs), carry out computations (including machine learning attacks or other precomputations for Phase 2 below), and fabricate physical systems (including special simulation devices and attempted physical clones of P). At that, $\mathcal{A}$ is only being limited by his own technological capabilities and equipment. At the end of Phase 1, $\mathcal{A}$'s physical access to P is ceased.

Phase 2: Response Prediction. A challenge C_i is drawn uniformly from the challenge space of P, and is given to $\mathcal{A}$. Within time t_{Pred}, $\mathcal{A}$ must output a "response prediction" $\hat{R}_i$. $\mathcal{A}$ wins the game if this prediction is correct, i.e., if $\hat{R}_i = R_i$. Thereby the probability ϵ is taken over all of $\mathcal{A}$'s random actions during FastPredGame, including any employed random fabrication processes. □

Let us quickly comment on the features and approach of Definition 9. It partly follows earlier, game-based definitions of PUFs [2,58], once more trying to strike a balance between rigor and accessibility.

Its security game FastPredGame closely mimics the situation of adversaries in Complex PUF schemes, such as Scheme 10: Their task usually is to predict responses of the Complex PUF quickly in order to break security. This is captured in Phase 2 of the game. Prior to this, the adversary commonly has substantial time for preparing his attack in practice, as reflected in the game's Phase 1. Providing $\mathcal{A}$ with *all* internal state $\mathsf{CIS(P)}$ of the PUF, while still requiring that security shall be preserved, makes Complex PUFs a *secret-free* security concept in the sense of this paper.

The chosen length of Phase 1 (=one year) is to some extent arbitrary, but is motivated by the practical parameters around certain modeling attacks on PUFs [57]; if anything, it overestimates the adversary's possibilities, leading to a yet stronger security notion. The reason for choosing a concrete length of Phase 1 (instead of assigning another parameter t_{Phase1}) is that it keeps the definition simpler. It also allows leaner and more concrete security analyses of any given, single Complex PUF candidates, as it avoids dependency on yet another parameter.

On a final note, Definition 9 is formulated relative to an adversary $\mathcal{A}$ and her/his/their individual technological capabilities and equipment. This will eventually lead to statements of the form "if a certain PUF is $(\epsilon, t_{\mathsf{Pred}})$-*Complex with respect to an adversary* $\mathcal{A}$, *also Scheme* X *is secure with respect to the same adversary* $\mathcal{A}$ ". Such relative formulations appear perfectly acceptable, however, following the standards of traditional, reductionist cryptography [24]. At the same time, such a "relative" definitional approach keeps our framework more flexible; please also compare our discussion subsequent to Definition 2 in this context.

4.2 Implementation of complex PUFs

From a fundamental perspective, the existence of Complex PUFs is substantiated by the fact that the simulation of certain physical systems can be laborious, sometimes even infeasible in practice (see, for example, [19]). Let us try to translate this idea specifically to a PUF-context. It is long known [57] that we can write a PUF's responses R_i as a function of the applied challenge C_i and the complete internal state $\mathsf{CIS(P)}$ of the PUF:

$$R_i = \mathsf{F_{PUF}}(C_i, \mathsf{CIS(P)})$$

Following this equation, the decisive point that distinguishes most standard PUFs from Complex PUFs is the higher computational complexity of the function $\mathsf{F_{PUF}}$. For example, $(\epsilon, t_{\mathsf{Pred}})$-Complex PUFs require an $\mathsf{F_{PUF}}$ that cannot be computed or emulated in any (possibly physical) way by the

adversary within the time period ϵ, even if the complete internal state $\mathsf{CIS(P)}$ of the PUF is known.

But which PUFs possess this property? A *concrete implementation candidate* of a $(\epsilon, t_{\mathsf{Pred}})$-Complex PUF for reasonable values ϵ and t_{Pred} is the optical PUF of Pappu et al. [47]. Following the security analysis in [46,47], we argue in some greater detail in Appendix 1 that at least under somewhat optimistic assumptions, Pappu et al.'s optical PUFs could be regarded a (13.3%, 10 s)-Complex PUFs with respect to realistic adversaries $\mathcal{A}$. We will assume these values in the further course of this paper in several schemes (see also Sect. 4.3).

Other Complex PUF candidates include the optical constructions of [33,72], while silicon candidates are much harder to find. They may include certain analog Strong PUFs [13,15]; also so-called SIMPL Systems [65] or Public PUFs [4] could theoretically directly serve as Complex PUF, not utilizing their response simulatability (compare Sect. 5). On the other hand, Arbiter PUFs and their variants are definitely no Complex PUFs for reasonably large values of t_{Pred}. Readers can easily convince themselves heir simulation complexity simply is too low: Once the signal runtimes in the Arbiter PUF components are known, simulating the outputs merely requires simple numeric addition of these runtimes. This can be carried out in logarithmic depth at time scales comparable to the challenge–response behavior of the original PUF hardware.

4.3 Complex PUF-based identification

Which secrets do Complex PUFs induce in hardware when employed in cryptographic protocols? Again, we use remote identification between a prover and a verifier as our guiding example, this time in connection with an optical PUF. To be fully precise, this implies that three pieces of hardware are involved in our protocol: *(a)* The prover's hardware $\mathcal{H}_{\mathsf{P}}$, consisting of a measurement apparatus and computing device. *(b)* The verifier's hardware $\mathcal{H}_{\mathsf{V}}$, consisting of a measurement apparatus and computing device. *(c)* An optical Complex PUF, i.e., a passive, non-electronic, optical scattering token, which is sometimes measured in the apparatus of the prover, sometimes in the apparatus of the verifier.

The resulting identification scheme has a security parameter λ, as well as n envisaged future runs, and is detailed in the following:

Scheme 10 (Identification with $(\epsilon, t_{\mathsf{Pred}})$-Complex PUFs).

Set-Up Phase (also called R_0):

1. The verifier's hardware $\mathcal{H}_{\mathsf{V}}$ chooses $\lambda \cdot n$ random challenges $C_1, \ldots, C_{\lambda \cdot n}$.
2. $\mathcal{H}_{\mathsf{V}}$ applies these challenges $C_1, \ldots, C_{\lambda \cdot n}$ to the optical $(\epsilon, t_{\mathsf{Pred}})$-Complex PUF, and collects the corresponding responses $R_1, \ldots, R_{\lambda \cdot n}$.
3. The resulting $\mathsf{CRP\text{-}List}$ $(C_1, R_1), \ldots, (C_{\lambda \cdot n}, R_{\lambda \cdot n})$ is stored permanently in $\mathcal{H}_{\mathsf{V}}$ in nonvolatile memory.

Execution Phase (Run R_i, with $1 \leq i \leq n$)

1. The verifier's hardware $\mathcal{H}_{\mathsf{V}}$ randomly selects λ CRPs $(C_1^i, R_1^i), \ldots, (C_\lambda^i, R_\lambda^i)$ from the $\mathsf{CRP\text{-}List}$.
2. $\mathcal{H}_{\mathsf{V}}$ sends $C_1^i, \ldots, C_\lambda^i$ to the prover's hardware $\mathcal{H}_{\mathsf{P}}$.
3. $\mathcal{H}_{\mathsf{P}}$ applies these challenges to the optical $(\epsilon, t_{\mathsf{Pred}})$-Complex PUF, and measures the resulting responses, which we call $\bar{R}_1^i, \ldots, \bar{R}_\lambda^i$.
4. $\mathcal{H}_{\mathsf{P}}$ sends $\bar{R}_1^i, \ldots, \bar{R}_\lambda^i$ to $\mathcal{H}_{\mathsf{V}}$.
5. $\mathcal{H}_{\mathsf{V}}$ measures the time period t^* that has passed between sending $C_1^i, \ldots, C_\lambda^i$ and receiving $\bar{R}_1^i, \ldots, \bar{R}_\lambda^i$.
6. $\mathcal{H}_{\mathsf{V}}$ then applies the following *decision rule*: *If* the responses $\bar{R}_1^i, \ldots, \bar{R}_\lambda^i$ arrived fast enough, i.e., if

   $$t^* \leq t_{\mathsf{Pred}},$$

   and if all these responses $\bar{R}_j^i$ were correct, i.e., if

   $$\bar{R}_j^i = R_j^i \quad \text{for all } j = 1, \ldots, \lambda,$$

   then $\mathcal{H}_{\mathsf{V}}$ accepts the identification, otherwise not.
7. The CRPs $(C_1^i, R_1^i), \ldots, (C_\lambda^i, R_\lambda^i)$ are removed from the $\mathsf{CRP\text{-}List}$. □

Let us quickly comment that error tolerance can be achieved easily by relaxing the decision rule of $\mathcal{H}_{\mathsf{V}}$ [46,47] : For example, a previously fixed number of incorrect responses $\bar{R}_j^i$ may be allowed. Or each sent response $\bar{R}_j^i$ may be permitted to differ from the pre-recorded value R_j^i in a fraction of bits [46,47]. Often syndrome-free error correction in the form of image transformations (e.g., the Gabor transformation [46,47] and others [54]) is applied in this context.

Regarding the security of Scheme 10, the following heuristic analysis holds. Let us call the (symmetric and unidirectional) communication and processing latency between verifier and prover (and vice versa) t_{Lat}, and the prover's time for measuring a response of the employed Complex PUF t_{Meas}. Then, it is not too difficult to see that Scheme 10 works securely as long as

$$t_{\mathsf{Pred}} \;\geq\; 2 \cdot t_{\mathsf{Lat}} + \lambda \cdot t_{\mathsf{Meas}}. \tag{1}$$

The rationale behind Eq. 1 lies in comparing an adversary $\mathcal{A}$ who is closely located to the verifier, and who has latency $t_{\mathsf{Lat}} \approx 0$ in the most extreme case, to a remote honest prover. $\mathcal{A}$ can use the full response time of this prover, which is around $2 \cdot t_{\mathsf{Lat}} + \lambda \cdot t_{\mathsf{Meas}}$, for his fraudulent numeric simulation of the λ responses $R_1^i, \ldots, R_\lambda^i$. He can thereby parallelize his simulation task to λ computers, while the honest prover

needs to measure the λ responses in sequence. This leads to the necessary and sufficient condition formulated in Eq. 1.

When applying Pappu et al.'s optical PUF in Scheme 10 over the internet, one could use parameters $t_{\mathsf{Pred}} = 10$ s, $t_{\mathsf{Meas}} = 0.1$ s, and $t_{\mathsf{Lat}} \leq 1$ s. Given the (optimistically) estimated value of $\epsilon = 13.3\%$ from Appendix 1, a target cheating probability of smaller than 2^{-100}, and an assumed CRP-measurement rate of 10 CRPs per second by the prover's measurement apparatus in practice, users can safely choose the security parameter $\lambda = 35$. Equation 1 is then fulfilled, and Scheme 10 is both secure and practical, for these example values.

4.4 Secrets in complex PUF-based identification

Let us now return to our main topic: Which *secrets* do Complex PUFs induce in hardware? As before, we use the earlier Scheme 10 as concrete discussion example. Please recall that there are three pieces of hardware in the scheme: *(a)* The measurement apparatus and computing device of the prover. *(b)* The measurement apparatus and computing device of the verifier. *(c)* An optical Complex PUF, i.e., an optical scattering token, which is sometimes measured in the apparatus of the prover, sometimes in the apparatus of the verifier.

Our analysis of the secrets in these hardware pieces is slightly shorter than in earlier schemes:

- The *prover's hardware* $\mathcal{H}_{\mathsf{P}}$ does not contain any secrets during the entire scheme, i.e., during runs $\mathsf{R}_0, \ldots, \mathsf{R}_n$. It is **secret-free**.
- The *verifier's hardware* $\mathcal{H}_{\mathsf{V}}$ contains **permanent non-volatile secrets**: Namely the CRP-List put together during R_0 and stored during all the rest of the scheme in NVM. Furthermore, $\mathcal{H}_{\mathsf{V}}$ also contains **transient digital secrets** during R_0: Namely the digital challenges C_i that are transferred to the measurement apparatus, and the resulting digital responses R_i that are received from the measurement apparatus.
  Finally, $\mathcal{H}_{\mathsf{V}}$ contains various **non-permanent, volatile physical secrets** during R_0: Firstly, the optical challenges C_i that $\mathcal{H}_{\mathsf{V}}$ generates (i.e., the laser beam and its position and angle of incidence). Secondly, the resulting physical, analog responses R_i that hit the CCD-camera of $\mathcal{H}_{\mathsf{V}}$.
- The *optical Complex PUF* itself also contains **non-permanent, volatile physical secrets** during the set-up phase R_0. This includes, first of all, the optical challenges C_i in the form of a laser beam that hits the PUF. The reason why they should be considered a secret is as follows: Imagine that they are given at the end of R_0 to an adversary. Assume further that, following the standard and established attack model for Strong PUF protocols [59], this adversary has physical access to the PUF at least once after the set-up phase. Then the adversary can apply all these challenges himself to the PUF, obtain the responses, and may later impersonate the PUF (or its holder) in communication with the verifier. This makes these analog optical challenges a permanent, volatile physical secret according to our earlier definitions. Secondly, the optical responses that leave the PUF in the form of a complicated optical wave during the set-up phase obviously are secrets for the same reasons.
  During the entire execution phase, i.e., during the runs $\mathsf{R}_1, \ldots, \mathsf{R}_n$, however, the optical Complex PUF is **secret-free**. Please recall in this context that the challenge–response pairs that are implicitly present in the respective runs $\mathsf{R}_1, \ldots, \mathsf{R}_n$ are no secrets in the sense of Definition 2: Given to the adversary at the end of the respective run, as stipulated in Definition 2, they do not break the security of past or future runs of the identification scheme.

This means that Scheme 10 implements a remote identification protocol with a secret-free prover during the entire scheme, and a secret-free Complex PUF/ identification token during the entire execution phase. This is provably impossible in a purely mathematical or Turing-machine-based scenario—constituting a noteworthy achievement of optical PUFs in our opinion.

One interesting aspect of the scheme is that although Complex PUFs are generally secret-free, they still can induce secrets in *other* pieces of hardware—in this case, the permanently stored CRP-List in the hardware of the verifier. Furthermore, depending on the exact cryptographic protocol they are used in, even the Complex PUFs themselves may carry volatile physical secrets for short periods—in our case, the physically applied optical challenges and resulting physical optical responses. This confirms our earlier mantra that the notion of secret-free security should be ideally be considered in relation to a certain cryptographic scheme and its specific properties in order to be meaningful.

We feel the above analysis also demonstrates the practical relevance of our taxonomy and theoretical framework. For example, the secrets which optical PUFs contain in the set-up phase usually are often overlooked in existing analyses of Scheme 10. Still, they can be exploited as attack vectors, for example, in the so-called "bad PUF model" [59,85]: This attack model (among other things) assumes that Strong PUFs could be manipulated by adversaries, so that they somehow "store" or "record" all challenges that have been applied to them earlier. In the case of optical Strong PUFs, this could potentially be achieved by adding a very thin layer of a special chemical substance onto the PUF, which changes permanently upon illumination, but which is only visible under UV-light, say.[9] Such "challenge-logging" bad PUFs

[9] This is just an illustrating example, of course, and many other variants are possible. In general, "challenge-logging" bad PUFs are easier

then destroy the security of various protocols under standard attack models, as detailed in [54,59,85]. Standardized analyses following our framework would allow the identification of such secrets and associated attack vectors at early stages, however. Their routine application could therefore be valuable to system designers and users alike.

5 Secret-free security by SIMPLs

5.1 Definition of SIMPLs

Can both verifier *and* prover become secret-free in identification schemes? This fundamental question almost immediately leads to the concept of a SIMPL System (or just SIMPL) [65], which also has been proposed equivalently and independently under the name Public PUF (or PPUF) [4]. The acronym SIMPL stands for "SIMulation Possible, but Laborious."

In a nutshell, a SIMPL System S is a PUF that possesses a publicly known description Des(S) of its individual physical disorder and randomness. This description is published and known to any protocol participants and adversaries alike. It shall allow the numeric simulation of the correct response R_i to any challenge C_i via some public simulation algorithm SIM:

$$R_i = \mathsf{SIM}(C_i, \mathsf{Des}(\mathsf{S}))$$

At the same time, this simulation (as well as any other simulation or emulation) shall be notably slower than the real-time behavior of S. Saying this in yet other words, obtaining the response R_i via physical measurement of the original, unique SIMPL shall be detectably faster than producing the same response via *any* (possibly adversarial) numeric simulation or hardware emulation. This fact allows immediate identification applications, for example, where the person holding the unique SIMPL can identify by presenting the *correct* responses to randomly chosen challenges over a digital communication line *quicker* than anyone else.

Due to the inherent limitations of existing manufacturing methods, the SIMPL S may indeed remain realistically unclonable even if Des(S) is publicly known. For similar reasons, also the digital simulation/emulation of S may remain difficult, even if all internal details of S are public. The promise therefore is that time gap between measuring and simulating/emulating responses R_i can be upheld, even against adversaries who know Des(S) and SIM. This renders SIMPLs a secret-free primitive in our sense (compare also [63,64]).

The above, informal discussion triggers the following sequence of definitions (compare again [63–65]).

Definition 11 (*SIMPLs* [63–65]) A *SIMPL* S is a PUF which possesses:

- A very large number of possible challenges, ideally (but not necessarily) exponential in some system parameter, such as the physical size or mass of S.
- A publicly accessible challenge–response mechanism, meaning that everyone with physical access to the SIMPL and/or the hardware embedding it can unrestrictedly apply challenges to the SIMPL and read out the corresponding responses.
- A publicly known description Des(S), which describes the individual features of S to a level of detail that allows simulation of the challenge–response pairs of S.
- A publicly known simulation algorithm SIM, which can simulate the correct response R_i of the SIMPL S when given a challenge C_i and Des(S):

$$R_i = \mathsf{SIM}(C_i, \mathsf{Des}(\mathsf{S})). \quad \square$$

We comment that the description Des(S) is assumed to be specific and individual for each SIMPL system, while the simulation algorithm SIM should be applicable to all SIMPL systems of a certain, joint design or class. We stress that the above definition merely clarifies the basic functionality of SIMPLs; their security features are stipulated next (compare again [63]):

Definition 12 ($(\epsilon, t_{\mathsf{Pred}})$-*Secure SIMPLs*) Let S be a SIMPL with internal state CIS(S) and simulation algorithm SIM. S is called an $(\epsilon, t_{\mathsf{Pred}})$-*secure SIMPL (or just an* $(\epsilon, t_{\mathsf{Pred}})$-*SIMPL) with respect to an adversary* $\mathcal{A}$ if $\mathcal{A}$ has a probability of at most ϵ to win the following game:

FastPredGame(S, $\mathcal{A}$, CIS(S), Des(S), SIM, t_{Pred}):

Phase 1: Preparation. $\mathcal{A}$ is given the binary strings CIS(S), Des(S) and SIM and physical access to S for one year. Throughout this period, $\mathcal{A}$ may conduct physical measurements on S (including determination of CRPs), carry out computations (including machine learning attacks or precomputations), and fabricate physical systems (including special simulation devices and attempted physical clones of S), only being limited by his own technological capabilities and equipment. At the end of Phase 1, $\mathcal{A}$'s physical access to S is ceased.

Phase 2: Response Prediction. A challenge C_i is drawn uniformly from the challenge space of S, and is given to $\mathcal{A}$.

to realize for electrical PUFs with a digital challenge–response interface: Maliciously adding a challenge-logger behind that interface is especially hard to detect. But via the above approach, also challenge-logging optical PUFs appear realistic to us.

Within time t_{Pred}, $\mathcal{A}$ must output a "response prediction" $\hat{R}_i$. $\mathcal{A}$ wins the game if this prediction is correct, i.e., if

$$\hat{R}_i = R_i.$$

Thereby the probability ϵ is taken over $\mathcal{A}$'s random actions during FastPredGame. □

We comment that Definitions 9 and 12 follow the same technical and conceptual approach; please therefore compare our explicatory discussion in Sect. 4.1. Again, the secret-free nature of SIMPLs/PPUFs is captured by giving all internal state $\mathsf{CIS(S)}$ and SIM to the adversary, while requiring that security is still upheld. We comment that Definitions 9 and 12 imply that any $(\epsilon, t_{\mathsf{Pred}})$-SIMPLs is a $(\epsilon, t_{\mathsf{Pred}})$-Complex PUFs for the same adversary $\mathcal{A}$—but not necessarily vice versa.

5.2 Implementation of SIMPLs

Similar to Complex PUFs, the existence of SIMPLs/ PPUFs is generally supported by the fact that most physical systems are Turing simulatable, but not necessarily efficiently (let alone in real-time), as pointed out early by Feynman [19]. Still, their implementation is an act of balance: Their simulation complexity cannot straightforwardly be maximized, as in the case of Complex PUFs. Instead, it must be finetuned and balanced to be large enough to prevent *quick* adversarial simulation, while still allowing response verification or *slow* simulation *at all* for honest users in practice.

Past research has yielded between one and two dozen SIMPL/PPUF implementation candidates, including [4,15,16,20,30,38,42,49–51,53,63–65,75] and references therein. This encompasses methods based on the random runtime delays in special digital circuits [4,42], nano-electronics and nano-circuits [49,51], analog circuits [15,16,20] and so-called cellular nonlinear networks [13,53], special-purpose circuits solving hard optimization problems on chip [38], special memory designs [53], or complex, disordered systems guided by differential equations, including optical systems [30,64,65,75].

We owe readers two illustrating (but not necessarily practical) didactic examples at this point, which they may keep in their minds throughout the rest of the paper to foster their own intuition. SIMPLs could, for example, be envisioned as optical PUFs whose complexity (i.e., number of scattering elements) has been gradually reduced to a regime where the optical PUF can be individually characterized and simulated, but where such simulation is still more time-consuming than the physical generation of an optical response by the PUF [65].

As second example, readers may consider a complex physical systems guided by differential equations (DEs) [30,65,75]. The challenge C_i could be applied by modifying the side/boundary conditions of the physical system and of its resulting DEs. The physical response R_i should ideally constitute some solution to these DEs. By simply inserting a candidate responses $\bar{R}_i$ into the DE, $\bar{R}_i$ could be checked for correctness quickly, then; but computing a correct response from scratch might be much harder. This approach, which has first been proposed in [65,75], [10] has the strong asset of allowing very quick response verification for the verifier. It thus reduces the computational burden of the verifier in Scheme 13, while upholding security and a high adversarial simulation time.

Finally, we would like to stress again that classical PUFs such as Arbiter PUFs are unsuited as SIMPLs/PPUFs, as their simulation and response prediction complexity is too low once their internal secrets have become known (compare Sect. 4.2). Good starting points for interested readers on SIMPL/ PPUF implementations could be [30,50,64,65,75].

5.3 SIMPL-based identification

Let us now detail the use of SIMPLs in our familiar benchmark application of remote identification [65]. Once, spelling out the details will serve the purpose of an accurate analysis later on.

Scheme 13 (Identification with $(\epsilon, t_{\mathsf{Pred}})$-SIMPLs [65])

Set-Up Phase (also called R_0):

1. The SIMPL S is fabricated, and SIM is derived.
2. SIM is permanently stored in $\mathcal{H}_{\mathsf{V}}$ in nonvolatile memory.

Execution Phase (Run R_i, with $1 \leq i \leq n$)

1. The verifier's hardware $\mathcal{H}_{\mathsf{V}}$ randomly selects λ challenges $C^i_1, \ldots, C^i_\lambda$.
2. $\mathcal{H}_{\mathsf{V}}$ sends $C^i_1, \ldots, C^i_\lambda$ to the prover's hardware $\mathcal{H}_{\mathsf{P}}$.
3. $\mathcal{H}_{\mathsf{P}}$ applies these challenges to the SIMPL, and measures the resulting responses, which we term $\bar{R}^i_1, \ldots, \bar{R}^i_\lambda$. $\mathcal{H}_{\mathsf{P}}$ sends $\bar{R}^i_1, \ldots \bar{R}^i_\lambda$ to $\mathcal{H}_{\mathsf{V}}$.
4. $\mathcal{H}_{\mathsf{V}}$ measures the time period t^* that has passed between sending $C^i_1, \ldots, C^i_\lambda$ and receiving $\bar{R}^i_1, \ldots, \bar{R}^i_\lambda$.
5. $\mathcal{H}_{\mathsf{V}}$ then applies the following *decision rule*: *If* the responses $\bar{R}^i_1, \ldots, \bar{R}^i_\lambda$ were sent fast enough, i.e., *if*

   $$t^* \leq t_{\mathsf{Pred}},$$

   and if all sent responses $\bar{R}_j$ were correct, i.e., if

   $$\mathsf{SIM}(C^i_j, \mathsf{Des(S)}) = \bar{R}^i_j \quad \text{for all } j = 1, \ldots, \lambda,$$

   then $\mathcal{H}_{\mathsf{V}}$ accepts the identification, otherwise not. □

[10] See, e.g., Section 6.3 of [65].

As earlier, some level of error tolerance can be achieved by appropriately relaxing the decision rule of $\mathcal{H}_{\mathsf{V}}$ on the correctness of the $\bar{R}_j^i$ in Step 5. Analogously to our discussion following Scheme 10 and also to Eq. 1, Scheme 13 is secure against an adversary $\mathcal{A}$ as long as

$$t_{\mathsf{Pred}} \geq 2 \cdot t_{\mathsf{Lat}} + \lambda \cdot t_{\mathsf{Meas}},$$

provided that an $(\epsilon, t_{\mathsf{Pred}})$-SIMPL/PPUF with respect to $\mathcal{A}$ was used. Again, we denote the (symmetric and unidirectional) communication/processing latency between verifier and prover (and vice versa) t_{Lat} at that, and the prover's time for measuring a response of the Complex PUF t_{Meas}.

In opposition to Scheme 10, the verifier in Scheme 13 needs to simulate the responses obtained from the prover in order to check their correctness. This consumes some time during the protocol. Pre-simulating the used λ CRPs prior to each protocol run R_i has been suggested in the literature [4,30] as countermeasure in order to speed up the protocol's execution time. This is possible, but also comes at a price: First of all, presimulation requires that the verifier knows that an identification protocol with a specific user and his SIMPL/PPUF is upcoming in the future. Secondly, presimulation unwantedly induces new secrets in the verifier's hardware, namely the pre-computed and stored CRPs. This makes the scheme no longer completely and permanently secret-free. (In passing, this again confirms our earlier mantra that also the use of secret-free primitives can induce secrets in hardware, depending on the exact scheme and the exact manner they are used. Only their "right" usage guarantees completely secret-free schemes and hardware.)

One alternative strategy to evade long computation times on the side of the verifier is the use of SIMPLs whose responses are hard to simulate from scratch, but which can be *verified* quickly. Disordered systems guided by differential equations, which have been sketched in [30,65,75], could be promising tools to this end. Such SIMPLs with quick response verification are somewhat hard to implement, but would clearly constitute an important breakthrough in the area.

5.4 Secrets in SIMPL-based identification

Let us now return to our main topic, namely the *secrets* that SIMPLs induce in remote identification protocols, more precisely in Scheme 13. Under the assumption that an $(\epsilon, t_{\mathsf{Pred}})$-secure SIMPLs is used, our analysis is delightfully brief:

- The *prover's hardware* $\mathcal{H}_{\mathsf{P}}$ is secret-free.
- Also the *verifier's hardware* $\mathcal{H}_{\mathsf{V}}$ is secret-free.

In other words, Scheme 13 is completely secret-free. From this angle, SIMPLs could hence be seen as a natural continuation and extension of both standard, digital public key cryptography [17], and of Complex PUFs: On the one hand, the use of public key techniques in remote identification [18] removes any secrets from the verifier's hardware, replacing them with a public key. Still, public key identification requires a secret key in the prover's hardware. On the other hand, the utilization of Complex PUFs avoids any secrets at the prover. But it still leads to digital secrets at the verifier. SIMPLs combine the best of both worlds: They remove secrets from *both* parties, the prover *and* the verifier.

We comment that all which Scheme 13 requires for remote identification is an authenticated, but public piece of information at the verifier. Intuitively, this appears as a minimal requirement: The verifier has to know at least *something* about the prover, otherwise the task of identification makes no sense and is ill-defined in the first place. Realizing this minimal requirement in a practically valid protocol marks a noteworthy achievement of PUF-like techniques and physical cryptography in our opinion. Once more, achieving the same is provably impossible to achieve via purely digital or Turing machine-like techniques.

6 Secret-free security by UNOs

6.1 Definition of UNOs

The third and last basic secret-free primitive which this manuscript shall discuss are so-called unique objects (UNOs). Let us start this section with some brief and distinguishing motivation for them.

Consider a small region of paper (0.01 mm^2, say), which is scanned by some external measurement apparatus like an electron microscope. This produces a high-resolution image $\overline{P}$ of its properties and structure [7]. However, due to limitations in existing 3D fabrication methods, adversaries could not fabricate a second piece of paper which, when being measured by the same electron microscope, would have the same properties $\overline{P}$ as the original [7]. This limitation even holds of adversaries would know $\overline{P}$. Our piece of paper hence must be considered secret-free and unclonable with respect to external measurement by the microscope.

The above phenomenon is an interesting security property, which is not yet directly captured by one of our earlier notions. This motivates the introduction of the complementary secret-free concept of a UNO below (compare [69]). As usual, will develop a semi-formal framework that captures their properties below, starting by a formal definition of the above-mentioned external measurement apparatuses.

Definition 14 (*Standard Measurement Apparatuses*) For the purposes of this paper, a so-called "Standard" Measurement Apparatus (SMA) is a device M that takes a physical object O as input, and produces a binary string $\mathsf{Prop}_{\mathsf{M}}(\mathsf{O})$ as output, which is called the *properties of* O *(upon measurement with* M*)*. We idealize that the measurement process executed by a SMA M shall leave the input object O unaltered. Furthermore, we stipulate that a SMA M shall be mass-producible, i.e., it shall be possible to fabricate an arbitrary number of *specimens* $\mathsf{M}', \mathsf{M}'', \ldots$ that possess the same input-output behavior.

Given these preparations, we can now define unique objects.

Definition 15 (*Unique Objects*) Let M be an SMA with complete internal state $\mathsf{CIS}(\mathsf{M})$, O be a physical object with complete internal state $\mathsf{CIS}(\mathsf{O})$ and properties $\mathsf{Prop}_{\mathsf{M}}(\mathsf{O})$, and let $\mathcal{A}$ be an adversary. O is called a *unique object (UNO) with unique properties* $\mathsf{Prop}_{\mathsf{M}}(\mathsf{O})$*with respect to* $\mathcal{A}$*and* M if it is practically infeasible for $\mathcal{A}$ to win the following game:

CloneGame$(\mathsf{O}, \mathsf{M}, \mathcal{A}, \mathsf{Prop}_{\mathsf{M}}(\mathsf{O}), \mathsf{CIS}(\mathsf{O}), \mathsf{CIS}(\mathsf{M}))$:

$\mathcal{A}$ is given O, M, $\mathsf{CIS}(\mathsf{O})$, and $\mathsf{CIS}(\mathsf{M})$. Within one year, $\mathcal{A}$ must output two physical objects O_1 and O_2, thereby only being limited in his actions by his technological capabilities and equipment. $\mathcal{A}$ wins the game if

$$\mathsf{Prop}_{\mathsf{M}}(\mathsf{O}) = \mathsf{Prop}_{\mathsf{M}}(\mathsf{O}_1) = \mathsf{Prop}_{\mathsf{M}}(\mathsf{O}_2). \qquad \square$$

We stress that the original object O may be destroyed or altered in the above **CloneGame** in order to make the definition most general: One of the objects O_1 and O_2 may be, but need not be, equal to O. Please note that Definition 15 follows the same technical approach as Definitions 9 and 11, creating a coherent definitional framework for all secret-free primitives of in this paper. As before, providing all relevant information $\mathsf{Prop}_{\mathsf{M}}(\mathsf{O})$, $\mathsf{InIn}_{\mathsf{O}}$, and $\mathsf{InIn}_{\mathsf{M}}$ to adversaries, while still requiring security to be maintained, captures the *secret-free* character of the considered primitive, in this case UNOs.

One interesting feature of UNOs is that adversaries are *forced* to generate real, physical objects in their attacks in order to break security. In all our earlier definitions, it was already sufficient if adversaries *could numerically output* a certain response correctly (and/or fast enough). UNOs therefore for the first time enforce real, physical cloning for adversaries. This can substantially enhance attack resilience, especially against adversaries who have no direct access to sophisticated fabrication equipment or clean rooms, such as standard consumers or small, non-state-funded hacker groups.

Conceptually, the price to be paid is the presence of a trusted, external, mass-producible measurement apparatus on site, which can physically measures the UNOs. Remote protocols between provers and verifiers, such as Schemes 7, 10, and 13, are not intended in a UNO-context. Still, these features allow application to two societally extremely relevant problems: The unforgeable tagging of valuable items (see Sect. 6.3) and protection and management of digital rights (see Sect. 7.4).

6.2 Implementation of UNOs

From a theoretical perspective, the existence of UNOs is motivated by the fundamental and well-known asymmetry between *measuring* and *fabricating* a physical object with a given precision. Measuring is both more accurate and more cost effective, in 2D as well as in 3D [46]. We stress again that no other known primitive exploits this asymmetry more directly than UNOs, since their attack model in Definition 15 *forces* the adversary to *physically* clone the UNO itself (compare Sect. 6.1).

Without explicitly using the term UNO (and mostly without developing a general conceptual theory behind it), a surprisingly large number of researchers have proposed and re-invented the basic idea of UNOs ad hoc and independently. The thread starts as early as in the late 1960s [39], continues in the 1980s [3,23] and 1990s [10,27,71,74,83,86,87], just to surface again in the 2000s [7,9,11,14,28,56,70,89]. Historically speaking, this line of research arguably marks the birth of what we call now "physical cryptography." Paper as employed physical medium plays a predominant role, being complex and stable, but still an everyday structure. Different measurement methods have been proposed to extract its unique features, including lasers [7,70] and ordinary scanners [11]. Other implementation suggestions include optical fibers [9], as well as radio wave [14] and magnetic [10,39,83] UNOs.

Again, we would like to propose three didactic, conceptual UNO-examples to readers, which they can hold in their minds throughout this section. The first one is the above-mentioned, tiny paper surface (or any other suited disordered surface), when being scanned with a high-resolution microscope. The second is Pappu et al.'s optical PUF [47]: Under the assumption that even if a certain raw response (=interference pattern) is known, no object can be produced which generates exactly this pattern upon laser illumination with a fixed challenge C_0, it is also a UNO. As a third example, image a variant of Pappu et al.'s PUF, where the number of scattering centers is reduced extremely strongly, so that only 25 scattering particles are contained in the structure, and that there are no further relevant manufacturing variations, say. The 3D position of each single scattering particle can then be determined with single digit nm accuracy in less than one second by existing microscopic techniques [84], while positioning the 25 particles with the same accuracy in 3D is currently impossible [84]. Unclonability hence is upheld,

even if all information about the structure is known, making it a UNO. At the same time, this object is not complex enough to serve as Strong PUF, Complex PUF, or SIMPL/PPUF, differentiating these concepts from UNOs.

To yet better draw the demarcation line between PUFs and UNOs, we should also mention that XOR Arbiter PUFs by virtue of Definition 15 are no UNOs: First of all, they contain secrets, namely their runtime delays. Secondly, a fixed, small set of their CRPs cannot serve as unique properties: Recall that an adversary know the target unique properties in his attack, and that XOR Arbiter PUFs have a digital challenge–response interface. The adversary can hence produce an effective clone with the same interface, that simply stores all few target CRPs of the XOR Arbiter PUF (i.e., all its UPs), and digitally outputs them whenever needed over this interface. Similar considerations apply to SRAM PUFs, which are no UNOs in our sense as well. Finally, we would like to stress that in opposition to many PUFs, often non-electronic, cost-effective, everyday media (such as paper) can be used as UNOs; please compare above. This promises a particularly easy mass-market applicability of theirs.

6.3 UNO-based unforgeable item tags/labels

UNOs can be applied to the efficient, unforgeable "tagging" or "labeling" of items of value, including passports, bankcards, banknotes, pharmaceuticals, security–critical components, consumer products, and the like. This constitutes an extremely relevant scientific problem: In 2013, the value of counterfeit and pirated goods was estimated between $923 Billion and $1.13 Trillion [79], with associated wider economic and social costs of $737 to $898 Billion [79], and desastrous employment losses of 2 to 2.6 million jobs [79]. Similar figures have been reported by the OECD [80] or Interpol [14].

The tagging scheme below assumes three basic parties or entities: *(i)* The *item (of value)* with an attached, UNO-based *tag*. *(ii)* The *manufacturer* of the items of value, or some other trusted third party, who can generate digital signatures that certify the tags. To this end, the manufacturer holds a secret signing key SK. *(iii)* The *testing device*, which verifies the tags and items for their validity by direct physical inspection. To this end, the testing device possesses the public verification key PK corresponding to SK, and an inbuilt standardized measurement apparatus M. Comparing this to our earlier, remote identification schemes, the *items of value* plus *tags* constitute some equivalent of the *provers*, the *testing devices* some analog to the *verifiers*. Scheme 16 could hence also be interpreted as *on-site* identification.

Scheme 16 (Unforgeable Labels/Item Tags with UNOs [14,23,69])

Set-Up Phase (also called R_0):

1. A UNO O is fabricated, and its unique properties $\mathsf{Prop}_{\mathsf{M}}(\mathsf{O})$ are determined by a specimen M' of the SMA M.
2. Using his signing key SK, which is stored permanently in nonvolatile digital memory, the *manufacturer* creates a digital signature $\mathsf{DigSig}_{\mathsf{SK}}(\mathsf{Prop}_{\mathsf{M}}(\mathsf{O}))$.
3. The UNO O is attached to the *item of value*. Jointly with it, $\mathsf{Prop}_{\mathsf{M}}(\mathsf{O})$ and $\mathsf{DigSig}_{\mathsf{SK}}(\mathsf{Prop}_{\mathsf{M}}(\mathsf{O}))$ are stored permanently on the *item of value*, for example, via a 2D barcode. Taken together, O, $\mathsf{Prop}_{\mathsf{M}}(\mathsf{O})$ and $\mathsf{DigSig}_{\mathsf{SK}}(\mathsf{Prop}_{\mathsf{M}}(\mathsf{O}))$ constitute the *tag*.

Execution Phase (Run R_i, with $1 \leq i \leq n$)

1. The *item* with its *tag* are presented to a *testing device*. The latter holds the public verification key PK corresponding to SK in nonvolatile memory, and possesses its own specimen M'' of the SMA M.
2. Using PK and M'', the *testing device* applies the following *decision rule*: *If*

   $$\mathsf{DigSig}_{\mathsf{SK}}(\mathsf{Prop}_{\mathsf{M}}(\mathsf{O}))$$

   is valid, *and* if for the object $\overline{\mathsf{O}}$ on the *item* it holds that

   $$\mathsf{Prop}_{\mathsf{M}}(\mathsf{O}) = \mathsf{Prop}_{\mathsf{M}}(\overline{\mathsf{O}}), \tag{2}$$

   then the *testing device* accepts the *tag*, otherwise not. □

From a security perspective, scheme 16 possess some intriguing upsides: Firstly, it conveniently allows an *offline* verification of tags, without an online channel to a central authority. This preserves the privacy of verifiers, and also maintains the non-traceability of tagged items, such as banknotes. The *item of value* itself thereby acts as a store-and-forward channel, permanently carrying the binary unique properties and digital signature, together with the physical object O. Secondly, the scheme is particularly useful in the context of offshore fabrication and illegitimate overproduction [1]: The company headquarters or intellectual property holder can provide their digital signatures *remotely* for tagging all items. Distrusted fabrication sites will then not possess their own signing keys, but can be kept secret-free instead, which is a strong asset. Finally, the tag's digital signature can also certify some additional, item-related information, such as biometric features of a passport owner, monetary value of banknotes, or consumer product data. This makes the general approach of Scheme 16 broadly applicable.

We remark that in order to be practical, the scheme ideally requires UNOs with unique properties $\mathsf{Prop}_{\mathsf{M}}(\mathsf{O})$ of the length of a few kilobytes, and also with reasonably small measurement times. Alternatively, hashed values of $\mathsf{Prop}_{\mathsf{M}}(\mathsf{O})$ can be signed and contained in the tags, and compared in Equation 2. As before, some small error tolerance and deviation may be allowed in this process, possibly using helper data [47]. Furthermore, the UNOs should allow high-precision measurement by cost-effective, standardized measurement apparatuses.

6.4 Secrets in UNO-based item tags

Let us now return to the *secrets* which UNOs induce in Scheme 16. Again, the list is delightfully brief:

- The *item of value* and the *tag* are secret-free.
- The *testing device* is secret-free.
- However, the *manufacturer* holds classical keys. Furthermore, skipping some details, the processing of these classical keys in the manufacturer's (digital) hardware will consequentially induce various other secrets. This includes transient digital secrets, and, depending on the exact system architecture, probably also non-permanent, nonvolatile digital secrets.

Scheme 16, therefore, establishes another approach in which certain hardware systems are secret-free. One particular asset here is that the two most widespread components of the scheme — the *items/tags* and the *testing devices*, which may exist in millions or even billions of samples—are secret-free. They hence do not require intricate and costly protection, but can remain inexpensive. The *manufacturer*, on the other hand, does hold a secret.

But this secret can be much better protected in a single company headquarter than elsewhere. Furthermore, in the case of offshore fabrication of the UNOs and the to-be-tagged items, the manufacturer can create his signatures from a remote, well-protected environment when being given $\mathsf{Prop}_{\mathsf{M}}(\mathsf{O})$ by the fabrication site. This gives the manufacturer full control over all valid labels, while not needing to share the signing key with other parties or the local fabrication site abroad.

7 Extension: other secret-free schemes?

The previous sections all focused on remote or on-site identification schemes as potentially key-free or secret-free application. This established a consistent benchmark for our comparative analyses. This section now deals with possible extensions of secret-free security to other settings, and ultimately also with the reach and limits of secret-free methods in general.

On the one hand, we will see that identification is certainly not the only possible appliance of secret-free techniques, and that their spectrum is considerably larger. On the other hand, it will become clear that probably not every conceivable cryptographic task can be realized in a *completely* secret-free manner either. Our discussion below tries to shed light on these and other aspects in a compact style.

7.1 Secret-free message authentication

One fundamental task beyond identification that is well-suited for secret-free security is message authentication (MACing). A concrete protocol to this end based on SIMPLs/PPUFs has first been described in [63,65]; it bears some similarities with Scheme 13, and can achieve both secret-free provers and secret-free verifiers. It is relatively straightforward to imagine similar schemes based on Complex PUFs (even though this has never been described explicitly in the literature to our knowledge). Such protocols based on Complex PUFs would achieve secret-free provers during the execution phase, but no secret-free verifiers.

Interestingly, both types of protocols (i.e., MACin with Complex PUFs and with SIMPLs/PPUFs) are inherently interactive and time-bounded in their nature: The verifier has to communicate in several rounds with the prover online in order to test the authenticity of a given message. Furthermore, the entire communication needs to be completed within a certain time frame, which is related to the adversarial prediction time t_{Pred} of the employed Complex PUF or SIMPL/PPUF (please compare Schemes 10, 13, and Definitions 9, 11). This makes the schemes interesting also from a theoretical perspective: They belong to the very few known interactive and time-bounded message authentication methods. Interested readers are referred to [63,65] for further details on SIMPL/PPUF-based MACing.

7.2 Secret-free tamper detection and PUF-capsules

One of the earliest proposed applications of PUFs has been tamper detection [22,47]: To this end, an optical PUF (or any other PUF whose CRPs are sensitive to violations of its structural integrity) shall encapsulate a piece of vulnerable hardware. By measuring CRPs of this "PUF-capsule" from the inside, and by employing them in a similar protocol as Scheme 7, the capsule's integrity and the "non-tamperedness" of the hardware inside it may be verified remotely.

Employing *Complex PUFs* or *SIMPLs/PPUFs* (instead of standard PUFs) as capsules, this approach can be implemented with a secret-free prover during the execution phase (if Complex PUFs are used). Or even with secret-free provers

and verifiers (if SIMPLs/PPUFs are utlized); please compare Schemes 10 and 13 in this context. Both approaches would provide a novel, secret-free manner of protecting vulnerable endpoints in large communication networks, such as the internet of things.

7.3 Secret-free sensing and virtual proofs of reality

Another emerging application of secret-free techniques lies in the domain of secure sensors. Let us assume to this end that Complex PUFs or SIMPLs/PPUFs could be reliably made dependent in their responses on the value EnvVar of a certain environmental variable, such as temperature, humidity, and pressure, following the equation below:

$$R_i = F(C_i, \mathsf{CIS(P)}, \mathsf{EnvVar}). \tag{3}$$

Under these circumstances, their CRPs can *prove* the actual value of the environmental variable to remotely located parties. For this purpose, protocols similar to Schemes 10 and 13 can be employed, in which each PUF-response is also a function of the desired environmental variable, as in Equation 3. The verifier in the set-up phase first needs to collect a different CRP-list for each discrete value of the environmental variable within a certain range. These CRP-lists are later used in communication protocols to show that the environmental variable at the PUF-location has a certain value. This is the principle of so-called virtual proofs (VPs) of reality, which have been detailed in [56].

Such VPs can be made secret-free on the side of the prover by using suitable environment-dependent Complex PUFs. They can even be made secret-free for both the verifier and prover by using SIMPLs/PPUFs [56] whose responses depend on the desired environmental variable. Interestingly, VPs with secret-free provers in the execution phase have already been demonstrated in experiment for *(i)* the destruction of a physical object, and for *(ii)* the spatial distance of two objects, using optical Complex PUFs in both cases [56]. Also temperature sensors based on electrical PUFs have already been realized in practice [56]. Finally, and most recently, VP-based techniques have been suggested in nuclear weapons inspections [48], introducing secret-free methods also in this highly demanding scenario. This makes secret-free sensing not just a theoretical, but also a practically viable option.

7.4 Secret-free digital rights management

Another fruitful application for secret-free techniques are digital rights management and copy-protection of valuable digital content. The basic observation that enables these usages is that even certain storage media like CDs/DVDs possess a "unique fingerprint," i.e., unique and unclonable physical properties, on a sub-digital and analog level [28,89]. Remarkably, this holds even if these media store exactly the same digital content.

Building on this observation, any digital content on CDs/DVDs can be certified, using digital signatures in combination with the physically unique features of each individual CD/DVD [28,89]. The approach in the end bears similarities to the use of UNOs as unforgeable labels, as spelled out in Sect. 6; technically speaking, the only major difference is that (a hash value of) the protected digital content is included in the digital signature in Step 2 of Scheme 16. Customary CD/DVD-readers can then serve as widespread, inexpensive, but surprisingly accurate measurement apparatus/testing device, as demonstrated in [28,89]. Both the CDs/DVDs and CD/DVD-readers are *secret-free* in this approach.

7.5 Secret-free encryption and digital signatures?

Is completely secret-free encryption possible? While this would be desirable, two fundamental obstacles exist.

Firstly, encryption obviously requires long-term security against adversaries. This probably implies that the necessary time gap t_{Pred} between the real-world behavior and *any* adversarial simulation/emulation of the Complex PUF or SIMPL/PPUF must be huge: It would have to be directly related to the aspired long-term confidentiality, which is on the order of many years or decades. At the same time, the implementation of Complex PUFs and SIMPLs/PPUFs currently is already highly challenging for medium values of t_{Pred}, which lie on the order of seconds or minutes. This obviously constitutes a severe issue.

Secondly, and perhaps yet more fundamentally, any encryption hardware necessarily needs to receive the plaintext at some point and in some form as input; but this plaintext by definition constitutes a secret in any encryption scheme. At least while this plaintext is provided to (and present in) hardware, the hardware hence cannot be completely secret-free.

This does not necessarily mean that secret-free encryption was no interesting topic for future research: Reduced forms of "secret-freeness" could still be possible. It is interesting to investigate, for example, whether encryption merely based on physical, non-permanent and volatile secrets is realizable in practice; future research will eventually have to tell. For now, let us merely comment that recently, some very interesting suggestions in this direction have been made in the quantum world [82]; the corresponding papers are recommended to interested readers.

Not exactly identical, but at least related considerations apply to secret-free digital signatures. They seem hard to realize via known SIMPLs/PPUFs because of the required gap between aspired long-term security and currently possible values of t_{Pred}. On the other hand, digital signatures do not share the fundamental problem of encryption schemes that the signed text necessarily constitutes a secret in itself. Future

investigations will have to reveal to which extent secret-free digital signatures are realizable in practice.

8 Summary and outlook

8.1 Summary

Secret keys that are stored permanently in digital form in NVM represent a standard ingredient in contemporary security architectures—but at the same time constitute a routinely exploited attack target in any resulting security hardware. One central motivation behind the introduction of PUFs was to overcome this problem: As PUFs can *avoid* permanently stored digital keys in NVM, they seemingly could immunize hardware against key extraction. Despite this seminal idea, PUFs perhaps surprisingly have been the subject of manifold and diverse attacks over the last decade, however [29,44,57,60,77].

By considering the formal concept of a secret in hardware, one can easily pinpoint the single, unifying reason behind these attacks: While standard PUFs can avoid classical, permanently stored digital keys, they often induce other types of secrets in hardware, which can again be extracted by smart adversaries. All above-mentioned PUF-attacks [29,44, 57,60,77] indeed have in common that they first identify, then extract secrets from PUFs: Modeling attacks deduced the (secret) signal delays of Arbiter PUFs via machine learning analysis of their CRPs [57]; certain physical attacks accomplished the same via photonic emission analysis of a running Arbiter PUF circuit [77]; yet other physical attacks on SRAM PUFs invasively obtained the (secret) power-up states of the involved SRAM cells [29]; and so on, and so on.

One theoretical way out of this dilemma would be the development of completely secret-free hardware—but such a step is non-trivial, unfortunately. It is provably impossible within a purely digital context, and has to be achieved by moving deeper into the physical layer. Along these lines, our paper described three realistic potential physical primitives by which certain forms of secret-freeness can be achieved in certain applications: Namely Complex PUFs, SIMPLs/PPUFs, and UNOs. In appliances like identification, message authentication, tamper protection, secure sensing, and yet a few others, these primitives can be employed to make some parties (such as provers in identification protocols), sometimes even all involved parties (such as both provers and verifiers in identification), secret-free.

Wherever achievable, the central promise of secret-freeness is to guarantee security in a rather extreme attack model: Even if adversaries could hypothetically inspect a piece of secret-free hardware bit by bit, and atom by atom, and were able to learn any information that is present in any form and at any time within the hardware, the security of the applications and protocols built on this hardware would still be upheld. Complex PUFs, SIMPLs/PPUFs, UNOs, and any other secret-free techniques thus can achieve provable and innate security against any kind of physical probing, side channels, or even malware-based key-extraction. In those settings in which they are applicable, they can thus bring the perpetual battle between key extractors and key protectors to an unexpected halt: Namely by removing any secrets, and thus any adversarial targets, from hardware. This feature is not met by previously existing purely digital methods, and also not by standard PUFs, as detailed in this manuscript. It hence seems to constitute a promising new route in hardware design and hardware security.

Besides the above contributions, we developed the first definitional framework for secrets, together with a formal taxonomy of secrets in hardware. This taxonomy might prove useful beyond completely secret-free hardware systems: It could be routinely applied to classify future security hardware according to the type of secrets it contains. With certain exceptions and counterexamples (compare Footnote 4 on page 6), our taxonomy generally induces a mild and consistent "hierarchy of secrets" in hardware. It ranges from more dangerous and vulnerable secrets to more resilient and harder-to-extract ones. Its routine application could hence help to identify potential system vulnerabilities at very early design stages. Sections 3.2, 3.3, and 4.4 illustrate this mechanism: They all *theoretically* identify secrets that have in the meantime been extracted in *practical* and *concrete* PUF attacks. In principle, these could have been identified by a standard application of our taxonomy early on.

Finally, Table 1 on page 23 subsumes our main findings. It provides both a historic overview and concrete examples for (almost) all different types of secrets from our taxonomy.

8.2 Three caveats

It seems important to us to explicitly issue three caveats toward the end of this manuscript, trying to avoid certain misunderstandings.

The first potential misunderstanding concerns the hope that secret-free security would allow unconditionally secure systems, on which simply *no* attack vectors exist, neither now nor in the future. This is not the case. As laid out in this work, also the security of secret-free primitives depends on certain unproven assumption. Two examples include the presumed time gap between the real-world behavior of a Complex PUF or SIMPL/PPUF and any possible adversarial emulation. Or, secondly, the assumed physical unclonability of these primitives, including UNOs. These two assumptions may turn out valid or not; they may potentially be attacked and broken by sophisticated adversaries in the future, and then restored by yet new designs, etc. What secret-free primitives promise to deliver is something else: They can completely disable

one (of many) attack vectors, namely physical and malware-based key extraction. Given that this is one of the main attack routes on contemporary electronic hardware, this still seems a noteworthy accomplishment, however.

A second important point is that completely secret-free solutions presumably will not exist for every cryptographic task. It seems hard to imagine, for example, how entirely secret-free encryption could be realized: Recall that the plaintext necessarily constitutes a secret in this application, and that it arguably must be present in any encryption and decryption hardware at least once and for limited time periods. This implies that such hardware cannot be considered completely secret-free (please compare Sect. 7.5). Also the simplistic hope that desktop computers and mobile communication hardware could be made continuously secret-free in any everyday applications would appear overstated. What Complex PUFs, SIMPLs/PPUFs, and UNOs can achieve for us, however, is secret-free security in certain basic applications and protocols, such as remote identification, message authentication, sensing, tamper detection, or forgery-proof item tagging. Future research needs to investigate how the secret-freeness in these fundamental protocols can be exploited in, and uplifted to, more complex tasks and system architectures. We assume that probably hybrid systems will come into existence, in which secret-freeness is achieved for certain time periods or in certain exposed hardware parts only; our taxonomy of secrets from Definition 3 can help us to routinely classify and analyze such systems for their vulnerabilities.

A last misunderstanding we would like to avoid is that all standard PUFs would not be useful (anymore). This is certainly not the case, and our taxonomy in Definition 3 actually witnesses this fact: While complete secret-free security may be the ultimate goal to fight key extraction, subforms (such as those in our taxonomy) are interesting and beneficial, too. This certainly includes standard PUFs as example, which can avoid classical keys. As they are practically easier to realize than completely secret-free systems, and since they arguably have a yet larger application spectrum (for example, in the form of Weak PUFs), they can be combined with secret-free methods to form the above-mentioned hybrid systems, in which the overall practical attack surface against key extraction is minimized.

8.3 Differences between secret-freeness and unclonability

At first, the two concepts of unclonability and secret-freeness might appear all too closely related, perhaps even identical. Let us therefore clarify their relation and differences in this short paragraph. To start with, any PUF-like hardware that is secret-free is necessarily also physically and digitally unclonable by definition. That is, adversaries in practice cannot fabricate a physical clone with a challenge–response behavior indistinguishable from the original PUF, and they also cannot derive a computer program (=a digital clone) that simulates and predicts the PUF's responses numerically.

This does not imply the converse, though—and, in fact, this lacking converse implication differentiates the two notions of unclonability and secret-freeness: A practically secure, i.e., digitally and physically unclonable PUF, may still contain secrets. As an example, think of a Arbiter PUF variant like the Interpose PUF (iPUF) [45] with very large sizes, which is beyond the reach of current machine learning attacks, and where also physical extraction of secrets may be difficult. Such a structure would be hard to clone in practice, both physically and digitally, but still contains secrets, for example, the exact delays and runtimes of its internal signals. This illustrates that unclonability and secret-freeness are two differing concepts.

Looking at the same circumstance from yet another angle, the practical security of PUF-like primitives can have two reasons: *(i)* The primitive contains no secrets at all. *(ii)* The primitive does contain secrets, but these are infeasible to extract in practice. While the PUF-literature has largely focused on case *(ii)* in the past, we would like to put forward in this paper that realizing case *(i)* seems, first of all, realistically possible given the latest developments in the field. And, secondly, that it might even represent the yet more promising strategy for realizing sustainedly secure hardware than standard PUFs or the attempted shielding of classical keys.

8.4 Future work

We believe that various research opportunities, perhaps even some research program evolves from the presented material. On the theory side, yet more closely investigating the unclonability and inherently slow simulatability of Complex PUFs, SIMPLS/PPUFs, and other secret-free primitives seems intriguing. So would be advancing the logical foundations of secret-free security, formally proving statements like "secret-free provers in secure remote identification protocols are equivalent to the existence of Complex PUFs." Thirdly, identifying secret-free primitives beyond the ones listed in this paper would appear fascinating—or formally showing that no such other secret-free primitives exist!

On the commercial side, the optimized design, practical realization, and widespread deployment of secret-free hardware seem attractive targets. Just to name three concrete examples: The development of CMOS-compatible Complex PUFs or SIMPLs/PPUFs; of secret-free sensors for physical variables beyond temperature [56]; or of secret-free item tags/labels which combine utmost measurement precision and security with cost efficiency and stability; all represent seminal and well-achievable goals. This paper is written in the hope to inspire and foster such activities.

Table 1 Resulting secrets at prover and verifier when different primitives are used for identification protocols, unforgeable labels, or for secure sensing/virtual proofs of reality. Historically, the table illustrates the gradual development from hardware containing classical keys to secret-free systems. Many of the listed features provably cannot be achieved in a purely digital setting, but inherently require physical approaches or primitives instead

Cryptographic Task/ Protocol	Employed Primitives/Keys	Resulting Secrets at Verifier (See Definitions 3, 4, 5)	Resulting Secrets at Prover (See Definitions 3, 4, 5)
Standard symmetric identification [1]	Secret digital keys stored permanently in NVM at prover and verifier	Permanent non-volatile digital secrets	Permanent non-volatile digital secrets
Standard asymmetric identification [1]	Private *(resp., public)* digital keys stored permanently in NVM at prover *(resp., verifier)*	**Secret-free**	Permanent non-volatile digital secrets
Weak PUF-based symmetric identification *(Scheme 6)*	SRAM PUFs at prover. Secret digital key stored permanently in NVM at verifier.	Permanent non-volatile digital secrets	Non-permanent volatile and transient digital secrets. Permanent non-volatile physical secrets. **Key-free.**
Weak PUF-based asymmetric identification *(see also [26])*	SRAM PUFs at prover. Public digital key stored permanently in NVM at verifier.	**Secret-free**	Non-permanent volatile and transient digital secrets. Permanent non-volatile physical secrets. **Key-free.**
Strong PUF-based identification *(Scheme 7)*	XOR Arbiter PUF at prover. Digital CRP-list stored permanently in NVM at verifier.	Permanent non-volatile digital secrets	Non-permanent volatile and transient digital secrets. Permanent non-volatile physical secrets. **Key-free.**
Complex PUF-based identification *(Scheme 10)*	Optical Complex PUF at prover. Digital CRP-list stored permanently in NVM at verifier.	Permanent non-volatile digital secrets	Non-permanent volatile physical secrets and **key-free** *(in set-up phase).* **Secret-free** *(in execution phase).*
Virtual proof of temperature *(compare Section 7.3, and Protocol 1 of [56])*	XOR Bistable Ring PUF at prover. Digital CRP-list stored permanently in NVM at verifier.	Permanent non-volatile digital secrets	Non-permanent volatile and transient digital secrets. Permanent non-volatile physical secrets. **Key-free.**
Virtual proof of distance *(compare Section 7.3, and Protocol 2 of [56])*	Optical Complex PUF at prover. Digital CRP-list stored permanently in NVM at verifier.	Permanent non-volatile digital secrets	Non-permanent volatile physical secrets and **key-free** *(in set-up phase).* **Secret-free** *(in execution phase).*
SIMPL-based identification *(Scheme 13)*	SIMPL/PPUF at prover. Public digital description (i.e., $\mathsf{Des}(\mathsf{S})$) stored permanently in NVM at verifier.	**Secret-free**	**Secret-free**
UNO-based unforgeable labels *(Scheme 16)*	UNO at prover/label. Public digital key stored permanently in NVM at verifier/testing apparatus.	**Secret-free**	**Secret-free**

Acknowledgements The author acknowledges support by BMBF Project QUBE and by BMBF Project PICOLA and by AFOSR Project FA9550-21-1-0039.

Funding Open Access funding enabled and organized by Projekt DEAL.

Appendix A: Analysis of Pappu et al.'s optical PUF as (ϵ, t_{Pred})-complex PUF

This section in the appendix discusses the suitability of Pappu et al.'s optical PUF construction [47] as (ϵ, t_{Pred})-Complex PUF for reasonable values of ϵ and t_{Pred}. We will simplify our analysis at times, and make a number of relatively optimistic assumptions, following the discussion in [47].

Let us start by deducing a somewhat realistic value of t_{Pred}. Pappu et al. estimate that given the entire internal structure of their optical PUF *(i.e., sizes and positions of scattering centers, etc.)*, in the most extreme case still up to 10^{26} computational operations are necessary to numerically predict responses with full exactness [47]. This estimate seems somewhat on the optimistic side; but if true, it would imply that the numeric computation of optical PUF responses may not even be practically possible at all in certain cases, and that t_{Pred} in practice is certainly much larger than 10 s, say. A guaranteed time margin of 10 s already suffices for many typical protocols in practice (please compare Scheme 10), however. While being aware that this may underestimate their real computational complexity, we thus for now and for our targeted application in identification protocols set $t_{\mathsf{Pred}} = 10$ s. Please recall in this context also that by definition, t_{Pred} is merely a *lower* bound for the adversarial response prediction time.

Let us next derive an adequate value for the parameter ϵ. Under the assumption that the 2400-bit keys derived from the multi-bit raw responses of Pappu et al.'s optical PUF are bitwise independent and all equally hard to predict (compare [46,47]), the probability of simply randomly guessing a response correctly would be as low as 2^{-2400} (i.e., considerably smaller than the standard guessing probability 1/2 for silicon PUFs with single-bit outputs). Direct random guessing of the response of the optical PUF hence is no viable strategy. Also no machine learning or other numerical approaches for deriving new, unknown responses from given CRPs are known in the case of Pappu et al.'s optical PUF. Let us therefore (again somewhat optimistically) assume that the above guessing probability of 2^{-2400} hence cannot be improved much by known numerical methods either.

If we again consider the (optimistic!) above assumption that numeric simulation of optical PUF-responses is too slow to matter in practice [46,47], the best adversarial approach for correctly predicting the optical PUF response to a randomly chosen challenge C_i in Phase 2 of the security experiment of Definition 9 therefore seems the following: *(i)* Collect as many CRPs as possible in Phase 1 of the security experiment of Definition 9, and *(ii)* hope that the posed challenge C_i then lies within this set of already known CRPs.

How large can said fraction of known CRP realistically become in Phase 1 under these assumptions? Pappu et al. estimate the number of decorrelated and independent CRPs of their optical PUF to be on the order of 2.37×10^{10} [46, 47]. Let us assume, once more somewhat optimistically from the perspective of honest users, that at most on the order of 10^2 CRPs can be measured per second by adversaries with access to the optical PUF, due to the necessary mechanical re-positioning of the laser and the frequency and data transfer limits of the CCD camera. Under this assumption, after one year of continuous CRP-measurements, an adversary will only have obtained a fraction of 13.3% of all CRPs. We hence estimate ϵ on the order of 13.3%.

Finally, it is important to stress that following the arguments of Pappu et al. [46,47], their optical PUF remains physically unclonable, *even if all of its internal structure is known to adversaries*. This is due to the current limitations of three-dimensional manufacturing techniques: It is assumed that they cannot position the scatterers in three dimension with the required precision [46,47]. This again guarantees the aspired secret-freeness of the optical PUF.

Assuming the correctness of the analysis in [46,47] and of our above example derivation, this would suggest that Pappu et al.'s optical PUFs could be regarded as (13.3%, 10 s)-Complex PUFs with respect to realistic adversaries $\mathcal{A}$. Finally, we stress again that our above exemplary analysis is rather optimistic in certain parts. It is to be understood as an example calculation that shall familiarize readers with our concepts, not so much as fully comprehensive or tight security analysis yet.

References

1. Anderson, R.J.: Security Engineering: A Guide to Building Dependable Distributed Systems. Wiley, New York (2010)
2. Armknecht, F., Maes, R., Sadeghi, A.R., Standaert, F.X., Wachsmann, C.: A formalization of the security features of physical functions. In: IEEE Symposium on Security & Privacy (2011)
3. Bauder, D.W.: An anti-counterfeiting concept for currency systems. Sandia National Labs, Albuquerque, NM, Technical Report, PTK-11990 (1983)
4. Beckmann, N., Potkonjak, M.: Hardware-based public-key cryptography with public physically unclonable functions. Inf. Hiding **2009**, 206–220 (2009)
5. Bennett, C.H., Brassard, G.: Quantum cryptography: public-key distribution and coin tossing. In: Proceedings of IEEE International Conference on Computers, Systems and Signal Processing, Bangalore, India, pp. 175–179 (1984)
6. Brzuska, C., Fischlin, M., Schröder, H., Katzenbeisser, S.: Physical unclonable functions in the universal composition framework. In: CRYPTO (2011)
7. Buchanan, J., Cowburn, R., Jausovec, A., Petit, D., Seem, P., Xiong, G., Atkinson, D., Fenton, K., Allwood, D., Bryan, M.: Forgery: fingerprinting documents and packaging. Nature **436**(7050), 475 (2005)
8. Chen, Q., Csaba, G., Lugli, P., Schlichtmann, U., Rührmair, U.: The bistable ring PUF: a new architecture for strong physical unclonable functions. In: HOST (2011)
9. Chen, Y., Mihcak, M.K., Kirovski, D.: Certifying authenticity via fiber-infused paper. SIGecom Exchanges **5**(3), 29–37 (2005)
10. Chu, M.C., Cheng, L.L., Cheng, L.M.: A novel magnetic card protection system. In: European convention on security and detection, pp. 207–211 (1995)
11. Clarkson, W., Weyrich, T., Finkelstein, A., Heninger, N., Halderman, J., Felten, E.: Fingerprinting blank paper using commodity scanners. In: IEEE Symposium on Security and Privacy (Oakland'09), pp. 301–314 (2009)
12. Clelland, C.T., Risca, V., Bancroft, C.: Hiding messages in DNA microdots. Nature **399**(6736), 533–534 (1999)
13. Csaba, G., Ju, X., Ma, Z., Chen, Q., Porod, W., Schmidhuber, J., Schlichtmann, U., Lugli, P., Rührmair, U.: Application of mismatched cellular nonlinear networks for physical cryptography. In: IEEE workshop on cellular nanoscale networks and their applications (CNNA), pp. 1–6 (2010)
14. DeJean, G., Kirovski, D.: RF-DNA: radio-frequency certificates of authenticity. In: CHES, pp. 346–363 (2007)
15. Deyati, S., Muldrey, B.J., Singh, A.D., Chatterjee, A.: Challenge engineering and design of analog push pull amplifier based physically unclonable function for hardware security. In: IEEE Asian test symposium (ATS), pp. 127–132 (2015)
16. Deyati, S.: Scalable algorithms and design for debug hardware for test, validation and security of mixed signal/rf circuits and systems. Ph.D. thesis, Georgia Institute of Technology (2017). https://smartech.gatech.edu/bitstream/handle/1853/58757/DEYATI-DISSERTATION-2017.pdf
17. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theory **22**, 644–654 (1976)
18. Feige, U., Fiat, A., Shamir, A.: Zero-knowledge proofs of identity. J. Cryptol. **1**, 77–94 (1988)
19. Feynman, R.P.: Simulating physics with computers. Int. J. Theor. Phys. **21**(6–7), 467–488 (1982)
20. Gassel, K.A.: Analog public PUF for hardware security. Master thesis, Georgia Institute of Technology (2018.) https://smartech.gatech.edu/bitstream/handle/1853/59961/GASSEL-THESIS-2018.pdf
21. Gassend, B., Clarke, D.E., van Dijk, M., Devadas, S.: Silicon physical random functions. In: ACM Conference on Computer and Communications Security (2002)
22. Gassend, B.: Physical random functions. MSc thesis, MIT (2003)
23. Goldman, R.N.: Non-counterfeitable document system. US-Patent 4,423,415. Publication date: 1983. Priority date: 1980. See https://patents.google.com/patent/US4423415A
24. Goldreich, O.: Foundations of Cryptography: Volume 2, Basic Applications. Cambridge University Press, Cambridge (2009)
25. Goorden, S.A., Horstmann, M., Mosk, A.P., Skoric, B., Pinkse, P.W.H.: Quantum-secure authentication of a physical unclonable key. Optica (2014)
26. Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: FPGA intrinsic PUFs and their use for IP protection. In: CHES, pp. 63–80 (2007)
27. Haist, T., Tiziani, H.J.: Optical detection of random features for high security applications. Opt. Commun. **147**, 173–179 (1998)
28. Hammouri, G., Dana, A., Sunar, B.: CDs have fingerprints too. In: CHES (2009)
29. Helfmeier, C., Boit, C., Nedospasov, D., Seifert, J.-P.: Cloning physically unclonable functions. In: HOST, pp. 1–6 (2013)
30. Herder, C.H.: Towards security without secrets. Ph.D. thesis, Massachusetts Institute of Technology (MIT) (2016)
31. Herder, C., Yu, M.D., Koushanfar, F., Devadas, S.: Physical unclonable functions and applications: a tutorial. Proc. IEEE **102**(8), 1126–1141 (2014)
32. Holcomb, D.E., Burleson, W.P., Fu, K.: Power-up SRAM state as an identifying fingerprint and source of true random numbers. IEEE Trans. Comput. **58**(9), 1198–1210 (2009)
33. Horstmeyer, R., Assawaworrarit, S., Rührmair, U., Yang, C.:: Physically secure and fully reconfigurable data storage using optical scattering. In: HOST, pp. 157–162
34. Kent, A.: Unconditionally secure bit commitment by transmitting measurement outcomes. Phys. Rev. Lett. **109**(13), 130501 (2012)
35. Kerckhoffs, A.: La cryptographie militaire. J. Sci. Mil. (1883)
36. Kumar, R, Burleson, W.: On design of a highly secure PUF based on non-linear current mirrors. In: HOST, pp. 38–43 (2014)
37. Leier, A., Richter, C., Banzhaf, W., Rauhe, H.: Cryptography with DNA binary strands. Biosystems **57**(1), 13–22 (2000)
38. Li, M., Miao, J., Zhong, K., Pan, D.Z.: Practical public PUF enabled by solving max-flow problem on chip. In: ACM/EDAC/IEEE Design Automation Conference (DAC) (2016)
39. Lindstrom, G., Schullstrom, G.: Verifiable identification document. US-Patent 3,636,318. Publication date: 1972. Priority date: 1968. See https://patents.google.com/patent/US3636318A
40. Lofstrom, K., Daasch, W.R., Taylor, D.: IC identification circuit using device mismatch. ISSCC **2000**, 372–373 (2000)
41. Maes, R., Verbauwhede, I.: Physically unclonable functions: a study on the state of the art and future research directions. In: Towards Hardware-Intrinsic Security, pp. 3–37. Springer (2010)
42. Majzoobi, M., Koushanfar, F.: Time-bounded authentication of FPGAs. IEEE Trans. Inf. Forensics Secur. **6**(3), 1123–1135 (2011)
43. Maurer, U.M.: Secret key agreement by public discussion from common information. IEEE Trans. Inf. Theory **39**(3), 733–742 (1993)
44. Nedospasov, D., Seifert, J.-P., Helfmeier, C., Boit, C.: Invasive PUF analysis. In: FDTC, pp. 30–38 (2013)
45. Nguyen, P.H., Sahoo, D.P., Jin, C., Mahmood, K., Rührmair, U., van Dijk, M.: The interpose PUF: secure PUF design against state-of-the-art machine learning attacks. IACR Trans. CHES (2019)
46. Pappu, R.: Physical one-way functions. Ph.D. thesis, Massachusetts Institute of Technology (2001)
47. Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical one-way functions. Science **297**, 2026–2030 (2002)
48. Philippe, S., et al.: The application of virtual proofs of reality to nuclear safeguards and arms control verification. In: 57th annual INMM meeting (2016)

49. Potkonjak, M., Meguerdichian, S., Nahapetian, A., Wei, S.: Differential public physically unclonable functions: architecture and applications. In: DAC (2007)
50. Potkonjak, M., Goudar, V.: Public physical unclonable functions. Proc. IEEE **102**(8), 1142–1156 (2014)
51. Rajendran, J., Rose, G.S., Karri, R., Potkonjak, M.: Nano-PPUF: a memristor-based security primitive. In: IEEE Computer Society Annual Symposium on VLSI (2012)
52. Ruehrmair, U., et al.: Method for security purposes. US Patent Application No. 13/250,534 (2012)
53. Rührmair, U., Chen, Q., Stutzmann, M., Lugli, P., Schlichtmann, U., Csaba, G.: Towards electrical, integrated implementations of SIMPL systems. WISTP (2010)
54. Rührmair, U., Hilgers, C., Urban, S., Weiershäuser, A., Dinter, E., Forster, B., Jirauschek, C.: Optical PUFs reloaded. IACR cryptology ePrint archive, report 2013/215 (2013)
55. Rührmair, U., Holcomb, D.E.: PUFs at a glance. In: DATE (2014)
56. Rührmair, U., Martinez-Hurtado, J.L., Xu, X., Kraeh, C., Hilgers, C., Kononchuk, D., Finley, J.J., Burleson, W.P.: Virtual proofs of reality and their physical implementation. In: IEEE Symposium on Security and Privacy (2015)
57. Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., Schmidhuber, J.: Modeling attacks on physical unclonable functions. In: ACM Conference on Computer and Communications Security (2010)
58. Rührmair, U., Sölter, J., Sehnke, F.: On the foundations of physical unclonable functions. IACR cryptology ePrint archive, report 2009/277 (2009)
59. Rührmair, U., van Dijk, M.: PUFs in security protocols: attack models and security evaluations. In: IEEE Symposium on Security and Privacy (2013)
60. Rührmair, U., Xu, X., Sölter, J., Mahmoud, A., Koushanfar, F., Burleson, W.: Power and timing side channels for PUFs and their efficient exploitation. IACR cryptology ePrint archive, report 2013/851 (2013)
61. Rührmair, U.: Oblivious transfer based on physical unclonable functions. In: TRUST (2010)
62. Rührmair, U.: Physical turing machines and the formalization of physical cryptography. IACR cryptology ePrint archive, report 2011/188 (2011)
63. Rührmair, U.: SIMPL systems as a keyless cryptographic and security primitive. In: Naccache, D. (Ed.) Cryptography and Security: From Theory to Applications. Lecture Notes in Computer Science, vol. 6805. Springer (2012)
64. Rührmair, U.: SIMPL systems, or: Can we build cryptographic hardware without secret key information? In: SOFSEM 2011. LNCS, vol. 6543. Springer (2011)
65. Rührmair, U.: SIMPL systems: on a public key variant of physical unclonable functions. IACR cryptology ePrint archive, report 2009/255 (2009)
66. Rührmair, U.: SoK: towards secret-free security. In: ASHES workshop at ACM CCS (2020)
67. Rührmair, U.: Towards secret-free security. IACR cryptology ePrint archive, report 2019/388 (2019). https://eprint.iacr.org/2019/388.pdf
68. Rührmair, U., van Dijk, M.: On the practical use of physical unclonable functions in oblivious transfer and bit commitment protocols. J. Cryptogr. Eng. (JCEN) **3**(1), 17–28 (2013)
69. Rührmair, U., Devadas, S., Koushanfar, F.: Security based on physical unclonability and disorder. In: Tehranipoor, M., Wang, C. (eds.) Introduction to Hardware Security and Trust. Springer, Berlin (2011)
70. Sharma, A., Subramanian, L., Brewer, E.A.: PaperSpeckle: microscopic fingerprinting of paper. In: ACM CCS, pp. 99–110 (2011)
71. Simmmons, G.J.: Identification of data, devices, documents and individuals. In: Annual International Carnahan Conference on Security Technology (1991)
72. Skoric, B., Schrijen, G.J., Ophey, W., Wolters, R., Verhaegh, N., van Geloven, J.: Experimental hardware for coating PUFs and optical PUFs. In: Security with noisy data, pp. 255–268. Springer, London (2007)
73. Skoric, B.: Quantum readout of physical unclonable functions. AFRICACRYPT (2010)
74. Smith, J.R., Sutherland, A.V.: Microstructure based indicia. In: Proceedings of the Second Workshop on Automatic Identification Advanced Technologies (1999)
75. Stutzmann, M., Csaba, G., Lugli, P., Finley, J.J., Jirauschek, C., Jaeger, C., Rührmair, U.: Towards electrical, integrated implementations of SIMPL systems. European Patent Application EP2230794 A3. Priority date: March 16, 2009. See https://patents.google.com/patent/EP2230794A3
76. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: Design Automation Conference, pp. 9–14 (2007)
77. Tajik, S., Dietz, E., Frohmann, S., Seifert, J.-P., Nedospasov, D., Helfmeier, C., Boit, C., Dittrich, H.: Physical characterization of arbiter PUFs. In: CHES, pp. 493–509 (2014)
78. Tehranipoor, M., Wang, C. (eds.): Introduction to Hardware Security and Trust. Springer, Berlin (2011)
79. The Economic Impacts of Counterfeiting and Piracy—Executive Summary. International Chamber of Commerce BASCAP and INTA Frontier Reports (2017). https://iccwbo.org/publication/economic-impacts-counterfeiting-piracy-report-prepared-bascap-inta/
80. Trade in Counterfeit and Pirated Goods: Mapping the Economic Impact. Organisation for Economic Co-operation and Development (OECD) (2016). See also: http://www.oecd.org/gov/risk/trade-in-counterfeit-and-pirated-goods-9789264252653-en.htm
81. Tuyls, P., Schrijen, G.-J., Skoric, B., van Geloven, J., Verhaegh, N., Wolters, R.: Read-proof hardware from protective coatings. CHES **2006**, 369–383 (2006)
82. Uppu, R., Wolterink, T.A.W., Goorden, S.A., Chen, B., Skoric, B., Mosk, A.P., Pinkse, P.W.H.: Asymmetric cryptography with physical unclonable keys. Quantum Sci. Technol. **4**, 045011 (2019)
83. Vaidya, A.W.: Keeping card data secure at low cost. In: European Convention on Security and Detection, pp. 212–215 (1995)
84. van den Broek, B., Ashcroft, B., Oosterkamp, T.H., van Noort, J.: Parallel nanometric 3D tracking of intracellular gold nanorods using multifocal two-photon microscopy. Nano Lett. **13**(3), 980–986 (2013)
85. van Dijk, M., Rührmair, U.: Physical unclonable functions in cryptographic protocols: security proofs and impossibility results. Cryptology ePrint archive, report 228/2012 (2012)
86. van Renesse, R.L.: 3DAS: a 3-dimensional-structure authentication system. In: European Convention on Security and Detection, pp. 45–49 (1995)
87. van Renesse, R.L.: Optical Document Security. Artech House, Boston (1998)
88. Vijayakumar, A., Kundu, S.: A novel modeling attack resistant PUF design based on non-linear voltage transfer characteristics. In: DATE, pp. 653–658 (2015)
89. Vijaywargi, D., Lewis, D., Kirovski, D.: Optical DNA. In: Financial Cryptography, pp. 222–229 (2009)
90. Yao, A.C.C.: Classical physics and the Church–Turing Thesis. J. ACM (JACM) **50**(1), 100–105 (2005)