Towards Secret-Free Security

Ulrich Rührmair
LMU München
80333 München, Germany
ruehrmair@ilo.de

***Abstract*—While digital secret keys appear indispensable in modern cryptography and security, they also routinely constitute a main attack point of the resulting hardware systems. Some recent approaches have tried to overcome this problem by *simply avoiding* keys and secrets in vulnerable systems. To start with, physical unclonable functions (PUFs) have demonstrated how "classical keys", i.e., permanently stored digital secret keys, can be evaded, realizing security devices that might be called "classically key-free". Still, most PUFs induce certain types of *physical secrets* deep in the hardware, whose disclosure to adversaries breaks security as well. Examples include the manufacturing variations that determine the power-up states of SRAM PUFs, or the signal runtimes of Arbiter PUFs, both of which have been extracted from PUF-hardware in practice, breaking security. A second generation of physical security primitives, such a SIMPLs/PPUFs and Unique Objects, recently has shown promise to overcome this issue, however. Perhaps counterintuitively, they would enable completely *"secret-free"* hardware, where adversaries might inspect *every* bit and atom, and learn *any* information present in *any* form in the hardware, without being able to break security. This concept paper takes this situation as starting point, and categorizes, formalizes, and surveys the currently emerging areas of *key-free* and, more importantly, *secret-free security*. Our treatment puts keys, secrets, and their respective avoidance into the center of the currently emerging physical security methods. It so aims to lay the foundations for future, secret-free security hardware, which would be innately and provably immune against any physical probing and key extraction.**

I. Introduction

A. Motivation and Overview

At the latest since the introduction of Kerckhoffs' principle in the 19th century [32], the notion of cryptographic security has always been intimately tied to the concept of a secret key. Many traditional tasks, such as identification, message authentication, or decryption, were considered possible solely under the assumption that the executing party had permanently stored a digital number in her hardware that was, and remained, unknown to adversaries. Such a number classically was referred to as a *"secret key"*.

It is not too difficult to see that in a purely Turing machine based view of computer security, such keys are indeed indispensable: If Alice is modeled as Turing machine, and if she wants to identify herself unambiguously to others, some *"part"* of her Turing program or tape must be unknown to impersonators. This part automatically constitutes a *secret key* of the identification scheme, then. Following similar arguments, the presence of secret keys in computer hardware seemed inevitable, and was accepted as a sheer necessity over decades.

Unfortunately, such a purely mathematically oriented view of matters brings along some well-known implementation issues: Effectively safeguarding keys in real-world devices still constitutes a major problem in computer and hardware security. Both malware attacks and physical extraction have proven highly efficient over the years [1]. The problem is exacerbated by the increasing level of mobility and connectivity of today's devices, their lightweight and often resource-constrained nature, and the complexity of operating systems and concurrently running programs and apps. This arguably turns permanently stored, digital secret keys into one of the Achilles heels of modern security hardware.

The above situation motivated the introduction of physical unclonable functions (PUFs) over a decade ago [40], [20]. Their central idea is to replace classical digital keys by complex physical structures, whose individual challenge-response behavior depends on unique and unclonable manufacturing variations. In this fashion, PUFs can individualize and identify hardware without non-volatile memory or permanent digital keys on board [20], [41], [29], and can also be employed in various advanced cryptographic protocols [49], [7]. They thus realize some initial, mild forms of *key-free security*. However, standard silicon PUFs cannot completely evade any sort of *"secrets"* in hardware yet: For example, the power-up states of SRAM PUFs [29], [23], or the individual runtime delays in Arbiter PUFs [20], still must remain *"secret"*, i.e., unknown to adversaries, for maintaining security. This has enabled a range of effective attacks in practice [55], [39], [26], [63], [57], which target and extract exactly these secrets.

A second class of physical security primitives, including so-called Unique Objects (UNOs) [51] and SIMPLs/PPUFs [46], [4], which has gone partly unnoticed by the broader public, recently shows promise to complete the endeavor originally started out by PUFs, however. Implemented successfully, they could realize hardware security in a surprisingly strong adversarial attack model: Even if the adversary was allowed to inspect the 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 scheme's execution, security would still be maintained. Such hardware is not just key-free, but might be called *secret-free* for obvious reasons. Being innately immune against any forms of key extraction, it could lead to a disruptive and groundbreaking change in the field — resolving the battle between key extractors and key protectors in a rather unexpected manner, namely by simply removing any secrets, and thus any targets, from vulnerable hardware.

B. Related Work

Key-free and secret-free security (in our sense) cannot be achieved in a purely mathematical or Turing machine setting, as observed above. All related works thus stem from the wider area of *"physical cryptography"*, which utilizes inherently physical phenomena for cryptographic and security purposes. The first well-known physical security primitive by which key-free, but usually no secret-free security can be achieved, are clearly classical PUFs [35], [41], [20] (see also [36], [51], [53], [27] and references therein). Follow-up works on SIMPL Systems (SIMPLs) [46] and Public PUFs (PPUFs) [4], judging from today's perspective, implicitly showed us how PUFs could be made secret-free. 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 SIMPL/PPUF primitives [46], [4].

Already before the advent of PUFs, a side strand of physical security research developed concepts that could be regarded as early, ad-hoc forms of secret-free UNOs in our terminology. These approaches date back to the late 1960s [34], and have resurfaced independently again in the 1980s [3], [22], 1990s [59], [44], [45], [58], [66], [11], [24], and 2000s [10], [15], [68], [25], [8], [12], [60]. It seems some of these methods went mostly unnoticed by the larger security community, though.

Finally, the use of physical assumptions in theoretical cryptography has a long tradition, including quantum [5], noise-based [38], DNA-based [13], [33], or relativistic cryptography [31]. Their motivation lies in replacing the standard, unproven computational complexity assumptions of mathematical cryptography by other, independent physical hypotheses.

None of the abovementioned publications explicitly makes the fundamental distinction between keys and secrets that underlies our paper, though. The historically first works in which the concept of secret-free security fully and unambiguously expressed (without using exactly this terminology) seem to be [47], [48] in 2011 and 2012 [1]. The first formal definition of a secret-free primitive (in this case a secret-free SIMPL) apparently appeared in 2012 [48] [2]. The only two other, later sources known to us are [27] and [28].

C. Our Contributions

The above situation makes our work the arguably first survey and concept paper that puts keys, secrets, and the possibility to avoid both, in its very center. To our knowledge, we develop the first definition of secrets, the first taxonomy of different keys and secrets in hardware, and the first definition of key-free and secret-free hardware. Furthermore, we develop the first coherent definitional framework for the three currently known secret-free primitives of Complex PUFs, SIMPLs/PPUFs, and UNOs. In doing so, we introduce the new notion of Complex PUFs as a gentle link between the key-free world of PUFs, and the secret-free world beyond. Furthermore, we introduce the concept of *internal information* of hardware and physical objects. Providing *any* internal information of *any* involved hardware to attackers, while still requiring that security should be upheld, turns out to be a generic technique for defining secret-free security that we develop in this paper.

Finally, as far as we know, we are the first to categorize existing security schemes with respect to our new taxonomy of keys and secrets. The results of our classification, which are summarized in Table I on page 17, illustrate the historic transition from key-free to secret-free security, serving as compact starting point for future research. Overall, to the best of our knowledge, this turns our work into the first concept paper and comprehensive survey of key-free and, perhaps more importantly, secret-free security.

D. Organization of This Paper

This manuscript is structured as follows: Section II provides a formal definition of secrets, a taxonomy of keys and secrets, and definitions of key-free and secret-free hardware. Section III discusses the complementary concept of isolation in hardware. Section IV analyzes the keys, secrets and isolation assumptions in typical Weak and Strong PUF scenarios. Sections V to VII in sequence detail three central secret-free security primitives: Complex PUFs, SIMPLs/PPUFs, and UNOs. Section VIII describes possible extensions of secret-free techniques. Table I on page 17 provides a compact overview of some of our main results.

II. A Definitional Framework for Secret-Free Security

A. Hardware Secrets

We will define some of our core concepts in this and the next subsections. As a preparatory step, it will be useful to stipulate the *standard security schemes* considered throughout the entire paper.

Definition 1 (Standard Security Schemes)**.** *Let $\mathcal{S}$ be a security scheme that fulfills the following properties:*

- *$\mathcal{S}$ involves a finite number of parties $\mathcal{P}_1, \ldots, \mathcal{P}_m$.*
- *$\mathcal{S}$ is implemented by a finite number of hardware systems $\mathcal{H}_1, \ldots, \mathcal{H}_k$ (with $k \geq m$).*
- *$\mathcal{S}$ consists of an initial set-up phase, which is termed R_0 for consistency reasons, and a subsequent execution phase with n sequential runs $\mathsf{R}_1, \ldots, \mathsf{R}_\mathsf{n}$.*

Under these prerequisites, we will call $\mathcal{S}$ a standard security scheme *in this paper.*

The above requirements imposed on standard security schemes are very mild, meaning that Definion 1 is hardly restrictive. The next definition now will now tell us something about secrets.

Definition 2 (Hardware Secrets)**.** *Let $\mathcal{S}$ be a standard security scheme. Then we call a piece of information s* a (hardware) secret *with respect to $\mathcal{S}$ if*

[1]See abstract and page 2 of [47], as well as abstract and page 2/3 of [48].

[2]Specifically, the string X used in Specification 1 on pages 4/5 of [48], which is known to attackers, is an early predecessor of our concept of *internal information* $\mathsf{InIn_P}$, $\mathsf{InIn_S}$, $\mathsf{InIn_O}$ used in the upcoming Definitions 9, 12, 15.

- *s is physically represented in arbitrary form (for example as content of volatile or non-volatile memory, physical properties of circuit components, physical characteristics of transient signals, including signal timing, shape or strength, internal physical states, structures, or configurations, etc.) in some hardware system $\mathcal{H}_i$ during some run $\mathsf{R_j}$ of $\mathcal{S}$ (where $\mathsf{R_j}$ may also be the set-up phase $\mathsf{R_0}$),*
- *and disclosure of s at the end of $\mathsf{R_j}$ allows adversaries to break some of the expected security features of at least one of the runs $\mathsf{R_1}, \ldots, \mathsf{R_n}$.*

We then also say that s is a secret in $\mathcal{H}_i$ (with respect to $\mathcal{S}$)*, or that $\mathcal{H}_i$* contains the secret s (during $\mathsf{R_j}$).

Let us briefly discuss the features of Definition 2, familiarizing us with its central ideas. First of all, 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 s 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 tell us, secrets need not always be binary numbers stored in digital 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 [50], 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. We comment that this approach does have some tradition also in theoretical computer science; compare, for example, the Church-Turing thesis [69], which is necessarily formulated in everyday language, and also others. Whilst 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 s *after* the execution of a certain run $\mathsf{R_j}$, not *during* it. The reason is relatively straightforward: Consider for illustration a standard 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 $\mathsf{PRF_K(N)}$ for a fixed pseudo-random function PRF, and returns this value to the verifier. Now, if the value $\mathsf{PRF_K(N)}$, which is 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 pass on $\mathsf{PRF_K(N)}$ to the verifier, achieving successful impersonation. A definitional framework that allows adversaries to learn potential secrets *right during* a run $\mathsf{R_j}$ would therefore formally turn $\mathsf{PRF_K(N)}$ into a *"secret"* of the identification scheme. While intuitively, and in any natural understanding of the term *"secret"*, it should not be considered as such. Definition 2 prevents this by stipulating that the attacker shall only learn potential secrets *after* the end of the respective run $\mathsf{R_j}$. In effect, this ensures that a secret is something deeper than just a response that is merely passed on or redirected. It must be a value that has a more profound impact, either backwards on the security of already completed runs, or on the future security of yet unstarted runs, in which new randomness is used.

B. A Taxonomy of Keys and Secrets

Recent physical security approaches, including PUFs, have demonstrated that there are different *"classes"* or *"types"* of secrets in hardware. This motivates the development of a *taxonomy of keys and secrets* below.

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

(i) Non-volatile digital secret*, or* **classical key***, if it is present in non-volatile digital memory of $\mathcal{H}_i$ at least once during $\mathcal{S}$.*
(ii) Volatile digital secret*, or* **volatile key***, if it is present in volatile digital memory of $\mathcal{H}_i$ at least once during $\mathcal{S}$.*
(iii) Transient digital secret *if it is present as the digital values of some transient digital signals in $\mathcal{H}_i$ at least once during $\mathcal{S}$.*
(iv) Digital secret *if is a classical key, volatile key, or a transient digital secret.*
(v) Physical secret *if it is not a digital secret, i.e., if it is never present as classical key, volatile key, or transient digital secret during $\mathcal{S}$.*
(vi) Non-volatile physical secret *if it is a physical secret that remains present in $\mathcal{H}_i$ if the power and other energy supplies for $\mathcal{H}_i$ are disabled.*
(vii) Volatile physical secret *if it is a physical secret that is lost once the power and other energy supplies for $\mathcal{H}_i$ are disabled.*

Let us quickly unveil the inner systematics of our definition. It mainly distinguishes keys and secrets along the following two dimensions: *(i)* Are they digital or physical? *(ii)* Are they volatile or non-volatile? We comment that this distinction implicitly introduces some *"hierarchy among secrets"* with respect to their resilience against physical attacks. To start with, digital secrets are usually more vulnerable than physical secrets, as information present in digital form is typically easier to extract than arbitrary analog characteristics. Furthermore, any non-volatile keys or secrets are commonly simpler to obtain for adversaries than their volatile counterparts: They remain present even if the power supply has been turned off, or in case the attacked hardware has been (partly) disassembled. In addition, non-volatile memory also usually shows stronger data remanence effects. Finally, extracting transient digital signals is commonly more difficult than adversarial read-out of digital memory.

In sum, this establishes some mild hierarchy amongst secrets, ranging from the most vulnerable secrets (i) to the most attack resilient ones (vii). While this hierarchy is by no means strict or binding, it does provide first guidelines for assessing the resilience of a given piece of hardware against probing and extraction attacks. Finally, our taxonomy stipulates standard secret keys (in non-volatile or volatile memory) as special cases of secrets; every such key is a secret, but not vice versa. This makes the notion of a *secret* a natural generalization of the known concept of a *secret key*.

C. Secret-Free Hardware and its Advantages

Whenever there are keys or secrets in hardware, adversaries will throw their wit and forces at extracting these. Just to name one example, even the non-classical keys and physical secrets in PUF-hardware have been extracted successfully in the past in practice [1], [39], [26], [63], [55], [57] (see also Section IV). This suggests the development of completely *"key-free"* and, more importantly, *"secret-free"* hardware. Solid definition of these two concepts will be the topic of this brief section.

Definition 4 (Key-Free and Secret-Free Hardware)**.** *Let $\mathcal{S}$ be a standard security scheme. A single hardware system $\mathcal{H}_i$ of $\mathcal{S}$ is called*

- classically key-free *if it contains no classical keys,*
- key-free *if it contains no classical and no volatile keys,*
- secret-free *if it contain no secrets at all,*

all with respect to $\mathcal{S}$.

Let us clearly work out the seminal advantages of key-free and secret-free security systems here in order to gain some momentum for the rest of the paper. To start with, (classically) key-free hardware obviously promises less vulnerability against standard invasive adversarial probing, as no secrets are present in (volatile or non-volatile) digital memory. Instead, transient digital signals or physical secrets have to be extracted by attackers, i.e., deeper levels of the circuit need to be accessed, or running signals need to be measured. At the same time, recent PUF-research has shown that attacks on such deeper levels are hard, but not impossible [55], [39], [26], [57], [63]. This means that key-free systems may improve attack resilience, but often cannot evade physical attacks completely.

Secret-free hardware takes this approach one step further: Even adversaries who can access *any* information that is present in hardware in *any* form during a security scheme, and may inspect every bit and every atom of the system, cannot break security. Secret-free hardware hence possesses provable, innate immunity against any type of adversarial probing or key-extraction, including malware.

Finally, we would like to highlight one other novelty of secret-free systems: They does not require any confidentiality in their set-up phases. The set-up can thus be accomplished under the eyes of the public, or in the presence of cryptographic adversaries, should this be necessary. This opens up new possibilities, for example in the context of nuclear weapons inspections, where set-ups for sensors and other monitoring equipment may need to be conducted in potentially hostile environments [43]. One concrete, practical example for a security scheme with non-confidential set-up phase in this context is Scheme 13.

III. A Glimpse at Isolation Assumptions

Every key and secret in hardware necessarily comes with an associated *isolation assumption*: The hypothesis that it will remain isolated and inaccessible to attackers. This holds both in terms of illegitimately reading as well as unauthorizedly (over-)writing or altering this key/secret. Following this insight, one could immediately formulate an associated isolation assumption for each of the secrets of Definition 2. However, simply *mirroring* the different types of secrets into *equivalent* isolation assumptions would not bring about additional value for our classification; it would merely portray the same facts and circumstances with yet other words. We therefore limit ourselves to one specific isolation characteristic that is orthogonal and independent of our previous discussion: So-called *activatable isolation*.

Definition 5 (Activatable Isolation)**.** *Let s be a secret in hardware $\mathcal{H}_i$ with respect to some standard security scheme $\mathcal{S}$. We say $\mathcal{S}$ assumes* activatable isolation for s in $\mathcal{H}_i$ *if:*

- *In the fabrication of $\mathcal{H}_i$, or in the set-up phase of $\mathcal{S}$, the secret s can be obtained/read from, or written into, $\mathcal{H}_i$ by certain third parties (e.g., by the manufacturers or owners of $\mathcal{H}_i$).*
- *After the fabrication or set-up phase, it is assumed that no parties other than $\mathcal{H}_i$ itself can obtain/read or write any parts of s from or into $\mathcal{H}_i$.*

Activatable isolation is a widespread and popular approach in many symmetric, secret-key based systems [1]. A standard implementation is triggering some irreversible physical change in the hardware, for example burning fuses [1]. Still, from a conceptual perspective, it is a significant assumption to make: Recall that the permanent isolation of keys and secrets can already be difficult to realize; but supposing that it can be discontinuously *"switched on"* or *"activated"* irreversibly is yet a stronger hypothesis. The problem here is that schemes with activatable isolation must not only deal with standard attacks, but also with adversaries who try to revert or circumvent the activated isolation mechanisms.

Avoiding activatable isolation is impossible in purely *classical, digital, symmetric* key based approaches: Either the key must be written into hardware $\mathcal{H}_i$ from the *"outside"*, e.g., during the fabrication process by the manufacturer, while it must be non-writable by external parties ever after. Or, if $\mathcal{H}_i$ generates the symmetric key *internally*, it must be made known to external parties (and hence obtained/read from $\mathcal{H}_i$) at least once, while external parties will not have access afterwards. In opposition to this, several physical primitives, including Strong PUFs, Complex PUFs, SIMPLs/PPUFs, and UNOs, can avoid activatable isolation. This is detailed over the next sections, and summarized at a glance in Table I on page 17.

IV. Keys, Secrets, and Isolation Assumptions in Standard PUFs

This section debuts our novel taxonomy and concepts at work: We will analyze the various keys, secrets, and isolation assumptions that arise from the use of typical Weak and Strong PUFs [53] in security schemes. Definition 2 implies that for a well-founded analysis, we need to pick a specific security scheme $\mathcal{S}$ first. Our benchmark scheme chosen here and throughout most of this paper is the remote identification of provers to verifiers over digital channels. As our discussion demonstrates, common silicon PUFs are able to realize certain forms of key-free, but no secret-free security in this application yet.

A. SRAM PUFs

Let us start by analyzing the keys, secrets, and isolation assumptions that result from the use of SRAM PUFs [29], [23], which are the most widespread Weak PUF [53] design, in symmetric identification. In the following scheme, we assume that an SRAM PUF is contained in the prover's hardware $\mathcal{H}_\mathsf{P}$.

Scheme 6: Symmetric Identification with SRAM PUFs [23], [29]

Set-Up Phase (also called R_0):

1) The prover's hardware $\mathcal{H}_\mathsf{P}$ measures the noisy power-up states of k SRAM cells of the SRAM PUF.
2) $\mathcal{H}_\mathsf{P}$ derives a stable secret key K and error-correcting helper data HD from these power-up states. The helper data HD is stored permanently in $\mathcal{H}_\mathsf{P}$.
3) $\mathcal{H}_\mathsf{P}$ gives K to the verifier's hardware $\mathcal{H}_\mathsf{V}$, where it is stored permanently.
4) The functionality of $\mathcal{H}_\mathsf{P}$,in which the secret key K may be given out to external parties like $\mathcal{H}_\mathsf{V}$ during the set-up phase, is irreversibly disabled.

Execution/Identification 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 N_i to the prover's hardware $\mathcal{H}_\mathsf{P}$.
2) $\mathcal{H}_\mathsf{P}$ derives the key K from the power-up states of its SRAM PUFs, using the helper data HD. It computes the value $\mathsf{PRF}_\mathsf{K}(\mathsf{N}_\mathsf{i})$ for some pseudorandom function PRF, and sends it remotely to the verifier's hardware.
3) The verifier's hardware checks the correctness of the received function value, using K. It accepts the identification if and only if the value is correct.

Applying our novel framework, we can identify the following *keys, secrets, and isolation assumptions* in Scheme 6:

- The *prover's hardware* $\mathcal{H}_\mathsf{P}$ is classically key-free, but contains volatile keys, namely the power-up states of the k SRAM cells. If $\mathcal{H}_\mathsf{P}$ is a fully digital system, it will also necessarily contain transient digital secrets: Firstly, the signals that communicate the noisy SRAM power-up states to the error correction mechanism within $\mathcal{H}_\mathsf{P}$. Secondly, the signals that transfer the error-corrected key K to the circuitry within $\mathcal{H}_\mathsf{P}$ that computes $\mathsf{PRF}_\mathsf{K}(\mathsf{N}_\mathsf{i})$. Thirdly, there will usually be further transient digital secrets in the circuitry computing $\mathsf{PRF}_\mathsf{K}(\mathsf{N}_\mathsf{i})$, their exact nature depending on the choice of PRF and its implementation. Finally, $\mathcal{H}_\mathsf{P}$ also contains non-volatile physical secrets, namely the physical manufacturing variations that determine the power-up states of the SRAM cells.
- The *verifier's hardware* $\mathcal{H}_\mathsf{V}$ contains a non-volatile digital secret, or classical key, namely K. $\mathcal{H}_\mathsf{V}$ is hence not even classically key-free.
- The scheme assumes activatable isolation in $\mathcal{H}_\mathsf{P}$.

Our analysis illustrates that in symmetric identification schemes, SRAM PUFs do evade classical keys in the prover's hardware. This feature provably cannot be achieved via standard, purely digital methods alone; it this constitutes a significant achievement of PUF-technology. On the other hand, the prover's hardware still contains volatile keys, transient digital secrets, and non-volatile physical secrets, and activatable isolation has to be assumed. Furthermore, the verifier's hardware must still store classical keys.

B. XOR Arbiter PUFs

Will common silicon Strong PUFs fare better? To start with, their large number of challenge-response pairs (CRPs), together with their unpredictable responses [53], allows a different type of identification protocol. In the upcoming Scheme 7 [41], the Strong PUF itself essentially functions as a pseudo-random function. Its CRPs can be sent in the clear over the digital communication channel, without deriving internal secret keys first. In order to concretize our analysis, we will assume that an *XOR Arbiter PUF* [62], [55], probably the most popular silicon Strong PUF [53] design, is contained in the prover's hardware $\mathcal{H}_\mathsf{P}$. The following, resulting scheme employs a security parameter λ, and has an envisaged number of n later runs.

Scheme 7: CRP-Based Identification with XOR Arbiter PUFs [41]

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 $\mathsf{CRP\text{-}List}$ (C_1, R_1), $\ldots$, $(C_{\lambda \cdot n}, R_{\lambda \cdot n})$.

Execution/Identification 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^i_1, R^i_1), $\ldots$, $(C^i_\lambda, R^i_\lambda)$ from the $\mathsf{CRP\text{-}List}$.
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 XOR Arbiter PUF, and sends the obtained responses $\bar{R}^i_1, \ldots, \bar{R}^i_\lambda$ to $\mathcal{H}_\mathsf{V}$.
4) $\mathcal{H}_\mathsf{V}$ checks if the received responses $\bar{R}^i_1, \ldots, \bar{R}^i_\lambda$ match the pre-recorded responses $R^i_1, \ldots, R^i_\lambda$ from the $\mathsf{CRP\text{-}List}$. It accepts the identification if and only if they

match within a certain, previously specified error margin *(for example on the number of incorrect responses, or on the number of allowed bitflips per response).*

5) The CRPs (C^i_1, R^i_1), ..., $(C^i_\lambda, R^i_\lambda)$ are removed from the $\mathsf{CRP\text{-}List}$.

Scheme 7 induces the following *keys, secrets, and isolation assumptions*:

- The *prover's hardware* $\mathcal{H}_\mathsf{P}$ is classically key-free, but contains volatile keys: Namely the digital content of the latches which act as *"arbiter elements"* at the end of the single Arbiter PUFs 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 [55]. For the same reason, the digital signals that enter the final XOR gate are transient digital secrets. Closely following Definitions 2 and 3, in its set-up phase (but not during its runs!), $\mathcal{H}_\mathsf{P}$ also contains another transient digital secret, namely the CRPs (C_1, R_1), ..., $(C_{\lambda \cdot n}, R_{\lambda \cdot n})$ that are collected via the digital challenge-response interface of the XOR Arbiter PUF to create the $\mathsf{CRP\text{-}List}$. The reason is that knowing these CRPs after the set-up $\mathsf{R_0}$, but before the runs $\mathsf{R_j}$, allows impersonation. Finally, $\mathcal{H}_\mathsf{P}$ contains non-volatile physical secrets: The manufacturing variations that cause the runtime delays in the XOR Arbiter PUF.
- The *verifier's hardware* $\mathcal{H}_\mathsf{V}$ contains a non-volatile digital secret, or classical key, namely the $\mathsf{CRP\text{-}List}$. $\mathcal{H}_\mathsf{V}$ is hence not even classically key-free.
- Scheme 7 does *not* assume activatable isolation in $\mathcal{H}_\mathsf{P}$.

This implies that if Strong PUFs are implemented by XOR Arbiter PUFs and comparable designs, and used in CRP-based identification, they induce a similar key and secret landscape as Weak PUFs in symmetric identification. While they can advantageously avoid activatable isolation at the prover, they are not able to realize secret-free security yet.

V. Secret-Free Security I: Complex PUFs

A. Basic Idea and Definitions

The perhaps easiest route to secret-free security is the novel concept of a *Complex PUF* introduced in this paper. Loosely speaking, Complex PUFs are a special sub-class of Strong PUFs [53], i.e., they must possess a large and practically inexhaustible CRP-space, a publicly accessible challenge-response interface, and responses that are difficult to predict numerically, even if many other challenge-response pairs are known [53]. On top of this, an $(\epsilon, t_\mathsf{AdvPre})$-*Complex PUF* shall have the following *additional property:* Even if the adversary knows the complete internal manufacturing variations and randomness of an $(\epsilon, t_\mathsf{AdvPre})$-Complex PUF P, 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 leads to the following two complementary definitions.

Definition 8 (Internal Information of Hardware). *Let* H *be a piece of hardware. Then the* internal information of H, *termed* $\mathsf{InIn_H}$, *is a bitstring that describes the general design and architecture of* H, *its memory content (if there is any), its random manufacturing variations, and any other physical variations resulting from intended individualization steps, such as burning fuses, or writing any physical patterns/keys into* H.

Definition 9 ($(\epsilon, t_\mathsf{AdvPre})$-Complex PUFs). *Let* P *be a PUF with internal information* $\mathsf{InIn_P}$. P *is called an* $(\epsilon, t_\mathsf{AdvPre})$-Complex PUF *with respect to an adversary* $\mathcal{A}$ *if* $\mathcal{A}$ *has a probability of at most* ϵ *to win the following, two-phase game:*

FastPredGame(P, $\mathcal{A}$, $\mathsf{InIn_P}$, t_AdvPre):

Phase 1: Preparation. *$\mathcal{A}$ is given the binary string $\mathsf{InIn_P}$ 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, and fabricate physical systems (including special simulation devices and attempted physical clones of P), 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_AdvPre, $\mathcal{A}$ must output a "response prediction" $\hat{R}_i$. He 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**.

Definition 9 partly follows earlier, game-based definitions on PUFs [56], [2], trying to strike a balance between rigor and accessibility. To start with, it uses concrete parameters instead of asymptotic concepts like polynomial time or negligible probabilities. This has the advantage of avoiding certain formal issues in PUF definitions [56], and also enables treating single PUFs (instead of infinite PUF families [56]). Furthermore, it allows a concrete (instead of merely asymptotic) analysis of PUF security properties, as in Section V-B).

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, as captured in Phase 2 of the game. Prior to this, the adversary commonly has substantial time for preparing his attack in practice, explaining the game's Phase 1. This game-based approach also allows us to exactly specify which information is available to the adversary in his attack. *Only* providing $\mathcal{A}$ with *all* internal information $\mathsf{InIn_P}$, while still requiring security to be upheld, makes Complex PUFs a *secret-free* security concept.

The chosen length of Phase 1 of one year is to some extent arbitrary, but is motivated by the effort of certain modeling attacks on PUFs [55]; if anything, it slightly overestimates the adversary's possibilities, leading to a stronger security notion. Choosing a concrete length of Phase 1 (instead of assigning another parameter t_Phase1) also keeps the definition simpler, and allows concrete security analyses of Complex PUF candidates (see Section V-B).

Finally, Definition 9 is formulated relative to an adversary $\mathcal{A}$, and his individual technological capabilities and equipment. This relative definitional approach will later lead to statements of the form *"if a certain PUF is $(\epsilon, t_{\mathsf{AdvPre}})$-Complex with respect to an adversary $\mathcal{A}$, also Scheme* X *is secure with respect to the same adversary $\mathcal{A}$ "* (see, for example, Sections V to Sections VII). Such relative formulations of security appear perfectly acceptable, though, and are also standard within traditional, reductionist cryptography [21].

B. Implementation

From a general viewpoint, the existence of Complex PUFs is motivated by the fact that the simulation of complex physical systems can be laborious, sometimes even practically infeasible [17]. A concrete *implementation candidate* is the optical PUF of Pappu et al. [41]. Since Complex PUFs are a new concept of this paper, we cannot refer to the literature for a full analysis of their properties, but conduct it ourselves in the sequel.

Let us start by deducing a realistic value of t_{AdvPre}. Pappu et al. estimate that given the entire internal structure of their optical PUF *(sizes and positions of scattering centers, etc.)*, in the most extreme case still up to 10^{26} computational operations are necessary in order to numerically predict responses with full exactness [41]. If true, this means that the numeric computation of optical PUF responses may not even be practically possible at all in certain cases, and that t_{AdvPre} in practice is certainly much larger than 10 sec, say. A guaranteed time margin of 10 sec already suffices for many typical protocols in practice (see Scheme 10), however. While being aware that this may strongly underestimate their computational complexity, we thus for now and for our targeted application in identification protocols set $t_{\mathsf{AdvPre}} = 10$ sec. Recall in this context also that t_{AdvPre} is a lower bound for the 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 [40], [41]), 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. The above probability of 2^{-2400} hence cannot be improved by such known methods either.

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 hence seems to collect as many CRPs as possible in Phase 1, and to hope that C_i then lies within this set of already known CRPs. How large can this fraction of known CRP realistically become in Phase 1? Pappu et al. estimate the number of decorrelated and independent CRPs of their optical PUF to be on the order of $2.37 \cdot 10^{10}$ [40], [41]. It seems reasonable to assume that at most on the order of 10^2 CRPs overall can be measured per second by adversaries with access to the optical PUF, due to the mechanical re-positioning of the laser, and the frequency and data transfer limits of the CCD camera. This means that after one year of continuous CRP-measurements, a realistic adversary $\mathcal{A}$ still 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. [40], [41], 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: They cannot position the scatterers in three dimension with the required precision far below the wavelength of the Also its simulation complexity remains high even in this case [40], [41]. This guarantees the aspired secret-freeness.

Assuming the correctness of the analysis in [40], [41], our above derivation illustrates that Pappu et al.'s optical PUFs can be regarded as $(13.3\%, 10 \text{ sec})$-Complex PUFs with respect to realistic adversaries $\mathcal{A}$.

Silicon candidates for Complex PUFs are much harder to identify. They include certain analog Strong PUFs [16], [14]. Furthermore, any so-called SIMPL Systems [46] or Public PUFs [4] could theoretically directly serve as Complex PUF, too (see Section VI), not utilizing their response verifiability. Arbiter PUFs and their variants are no Complex PUFs for reasonably large values of t_{AdvPre}, as their simulation complexity is simply too low: Once the runtimes in their components are known, simulating their outputs merely requires simple numeric addition of these runtimes.

C. An Example Scheme

Which effect do Complex PUFs have on the presence of keys and secrets in hardware? Again, we employ identification as our benachmark example, assuming that the prover's hardware $\mathcal{H}_{\mathsf{P}}$ carries an optical Complex PUF. The following scheme has security parameter λ and n envisaged future runs.

Scheme 10: CRP-BASED IDENTIFICATION WITH $(\epsilon, t_{\mathsf{AdvPre}})$-COMPLEX OPTICAL 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 $(\epsilon, t_{\mathsf{AdvPre}})$-complex optical PUF, and collects the corresponding responses $R_1, \ldots, R_{\lambda \cdot n}$.
3) The resulting $\mathsf{CRP\text{-}List}$ (C_1, R_1), …, $(C_{\lambda \cdot n}, R_{\lambda \cdot n})$ is stored permanently in $\mathcal{H}_{\mathsf{V}}$.

Execution Phase (Run $\mathsf{R_i}$, 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), …, $(C_\lambda^i, R_\lambda^i)$ 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 PUF, and determines the corresponding responses, which we call $\bar{R}^i_1, \ldots, \bar{R}^i_\lambda$. $\mathcal{H}_{\mathsf{P}}$ 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

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

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

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

then $\mathcal{H}_{\mathsf{V}}$ accepts the identification, otherwise not.
6) The CRPs (C^i_1, R^i_1), ..., $(C^i_\lambda, R^i_\lambda)$ are removed from the CRP-List.

Let us start our discussion by the straightforward comment that error-tolerance can be achieved easily by relaxing the decision rule of $\mathcal{H}_{\mathsf{V}}$: For example, a previously fixed number of incorrect responses $\bar{R}^i_j$ may be allowed. Or, each sent response $\bar{R}^i_j$ may be permitted to differ from the pre-recorded value R^i_j in a fraction of bits.

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 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

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

The rationale behind Equation 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^i_1, ..., R^i_λ. 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 of Equation 1.

When applying Pappu et al.'s optical PUFs in Scheme 10 over the internet, one can estimate $t_{\mathsf{AdvPre}} = 10$ sec, $t_{\mathsf{Meas}} = 0.1$ sec, and $t_{\mathsf{Lat}} \leq 1$ sec. Given the estimated value of $\epsilon = 13.3\%$ from earlier, and a target cheating probability of smaller than 2^{-100}, this means that users have to choose the security parameter $\lambda = 35$. Equation 1 is then fulfilled, and Scheme 10 is secure, for these values.

D. Keys, Secrets, and Isolation Assumptions

The last sections plausibilized the existence and also secure applicability of Complex PUFs in remote identification. Let us now return to our main topic: Which *keys, secrets, and isolation assumptions* do they induce in Scheme 10?

- The *prover's hardware* $\mathcal{H}_{\mathsf{P}}$ does not contain any secrets during the entire execution phase, that is, during any of the runs $\mathsf{R}_1, \ldots, \mathsf{R}_\mathsf{n}$. It merely contains volatile physical secrets in the set-up phase: Namely the physical, analog optical challenges C_i, which are externally applied to the PUF by the *verifier's* hardware $\mathcal{H}_{\mathsf{V}}$ during the set-up, and the resulting physical, analog responses R_i. *(Recall that the CRPs in Scheme 10 are* **not** *collected via a digital CRP-interface, which is a peculiarity of optical PUFs).*
- The *verifier's hardware* $\mathcal{H}_{\mathsf{V}}$ contains classical keys, namely the CRP-List.
- *No* activatable isolation is assumed in $\mathcal{H}_{\mathsf{P}}$.

In other words, Scheme 10 for the first time in this paper implements a secret-free prover during the execution phase! As observed already in Section I, this is impossible to realize in a purely mathematical or Turing-machine based scenario. It constitutes a first major achievement of physical cryptography in our opinion.

VI. Secret-Free Security II: SIMPLs

A. Basic Idea and Definitions

Can *both* verifier *and* prover become secret-free in identification schemes? If yes, the verifier must merely use some public information (not a secret CRP-List!) for checking the prover's responses. This leads to the concept of a SIMPL System (or SIMPL) [46], which independently has been suggested under the name Public PUF (or PPUF) [4].[3]

Loosely speaking, SIMPLs/PPUFs are PUFs that allow the *numerical verification* of the correctness of CRPs via some public, non-secret verification algorithm. At the same time, knowledge of this algorithm should not enable adversaries to *generate* correct CRPs at will and with arbitrary speeds: Otherwise, straightforward attacks become possible, such as impersonation in CRP-based identification (see Scheme 13). This means that a comparable asymmetry as in public key cryptography is immanent in SIMPLs/PPUFs, whence they may truly be regarded as an *asymmetric* or *public key variant* of PUFs [46], [4].

The upcoming definition clarifies their basic functionality.

Definition 11 (SIMPLs/PPUFs). *A SIMPL/PPUF* S *is a PUF that possesses an associated, public verification algorithm* $\mathsf{Ver_S}$*. Given an input of the form* $(C_i, \bar{R}_i)$, $\mathsf{Ver_S}$ *shall produce the following output:*

- 1*, if* $\bar{R}_i$ *is the correct response* R_i *of* S *to challenge* C_i*.*
- 0*, otherwise.*

Furthermore, S *is said to have:*

[3]SIMPLs and PPUFs are fully equivalent concepts, that have been introduced by independent groups around the same time. The first publicly available, timestamped document indeed used the term SIMPL system [61]. Also, the identification application that we describe in this paper has first been put forward in the context of SIMPLs [46], and so has the observation that quick verification instead of slow simulation might be a better suited implementation approach [46]. Finally, the fact that these structures may be entirely secret-free also has first been observed in the SIMPL-related literature [47], [48]. On the other hand, the name Public PUF [4] seems to some extent more intuitive. Furthermore, a strong share of implementation candidates have been published in the literature strand on Public PUFs (see, e.g., [42] and references therein). Exclusively picking one name over the other would thus seem unfair towards either of the involved groups. We will therefore use the fair and suggestive hybrid expression SIMPL/PPUF in this paper.

- Verification time t_{Ver} for a party $\mathcal{P}$ *if* $\mathcal{P}$ *can run* $\mathsf{Ver_S}$ *on any inputs* $(C_i, \bar{R}_i)$ *within time* t_{Ver}.
- Measurement time t_{Meas} for a party $\mathcal{P}$ *if* $\mathcal{P}$ *can physically measure the correct response* R_i *of* S *to any challenge* C_i *within time* t_{Meas}.

We comment that most existing works on SIMPLs/PPUFs do not postulate a verification algorithm, but an explicit simulation algorithm for their responses [46], [4]. However, as our discussion following Scheme 13 shows, such purely simulation-based approaches necessarily lead to some efficiency vs. security issues. Merely postulating a verification instead of simulation algorithm also creates a more general definition, since simulatability implies verifiability, but not vice versa. This motivates and explains our choice.

The security features of SIMPLs/PPUFs are detailed next.

Definition 12 (($\epsilon, t_{\mathsf{AdvPre}}$)-SIMPLs/PPUFs)**.** *Let* S *be a SIMPL/PPUF with internal information* $\mathsf{InIn_S}$ *and verification algorithm* $\mathsf{Ver_S}$. S *is called an* $(\epsilon, t_{\mathsf{AdvPre}})$-SIMPL/PPUF with respect to an adversary $\mathcal{A}$ *if* $\mathcal{A}$ *has a probability of at most* ϵ *to win the following, two-phase game:*

FastPredGame(S, $\mathcal{A}$, $\mathsf{InIn_S}$, $\mathsf{Ver_S}$, t_{AdvPre}):

Phase 1: Preparation. $\mathcal{A}$ *is given the binary strings* $\mathsf{InIn_S}$ *and* $\mathsf{Ver_S}$ *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, 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* $\mathcal{A}$. *Within time* t_{AdvPre}, $\mathcal{A}$ *must output a "response prediction"* $\hat{R}_i$. *He 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**.

Definitions 9 and 12 follow the same technical approach; please compare our explanatory discussion in Section V-A. Again, the secret-free nature of SIMPLs/PPUFs is captured by giving all internal information $\mathsf{InIn_S}$ and $\mathsf{Ver_S}$ to the adversary, while requiring that security still is upheld. The definitions imply that any $(\epsilon, t_{\mathsf{AdvPre}})$-SIMPLs/PPUFs is a $(\epsilon, t_{\mathsf{AdvPre}})$-Complex PUFs for the same adversary $\mathcal{A}$ — but not vice versa.

B. Implementation

From a fundamental viewpoint, the existence of SIMPLs/PPUFs is 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 [17]. Still, their implementation is an act of balance: Simulation complexity cannot simply be maximized, as in the case of Complex PUFs. Instead, it must be finetuned to be large enough to prevent *quick* simulation, while still allowing response verification (or *slow* simulation, see below) in practice.

Recent research has yielded around a dozen SIMPL/PPUF implementation candidates, including [42], [47], [48], [4], [46], [28], [37] and references therein. Almost all pursue the following conceptual approach: A SIMPL/PPUF S is fabricated, and its relevant manufacturing variations are characterized, either by direct physical probing, or by numerical analysis of many CRPs. This results in a *simulation algorithm* $\mathsf{Sim_S}$, by which correct responses can be simulated numerically. The verification algorithm $\mathsf{Ver_S}$ of Definitions 11 on input $(C_i, \bar{R}_i)$ then just needs to simulate the correct response by use of $\mathsf{Sim_S}$, and to compare it to its input $\bar{R}_i$. If S is sufficiently complex, response simulation will be possible, but laborious and slow, and the security features of Definition 12 will be met for reasonable values of t_{AdvPre}. This approach could be termed *Simulation-Based (SB) SIMPLs/PPUFs*.

Conceptually, there is an alternative strategy [46], [48], which we like to call *Quickly Verifiable (QV) SIMPLs/PPUFs*. Its idea is to exploit the fact that if cleverly designed, the SIMPL/PPUF might allow some rapidly computable predicate for response correctness, allowing a fast algorithm $\mathsf{Ver_S}$ that does *not* involve full response simulation from scratch. This approach has first been suggested in [46], [48] and continued in [28]. It is arguably harder to implement, but offers important efficiency vs. security advantages (see Section VI-C).

We owe readers two illustrating (but not necessarily practical) didactic examples, which they may keep throughout the rest of the paper for their own intuition. SB SIMPLs/PPUFs could 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 characterized and simulated, but where such simulation is still inevitably time-consuming [46].

QV SIMPLs/PPUFs, on the other hand, might be imagined as complex physical systems guided by differential equations (DEs) [61], [28]. 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. Arbiter PUFs are no SIMPLs/PPUFs, as their internal simulation complexity is too low (see Section V-B). Recommended starting points for further reading on SIMPL/PPUF implementation are [42], [28], [47].

C. An Example Scheme

Let us now detail the use of SIMPLs/PPUFs in our benchmark application of identification [46].

Scheme 13: CRP-BASED IDENTIFICATION WITH $(\epsilon, t_{\mathsf{AdvPre}})$-SIMPLS/PPUFS [46]

Set-Up Phase (also called $\mathsf{R_0}$):

1) The SIMPL/PPUF S is fabricated, and $\mathsf{Ver_S}$ is derived.

2) $\mathsf{Ver_S}$ is permanently stored in $\mathcal{H}_\mathsf{V}$.

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

1) The verifier's hardware $\mathcal{H}_\mathsf{V}$ randomly selects λ challenges $C_1, \ldots, C_\lambda$.
2) $\mathcal{H}_\mathsf{V}$ sends $C_1, \ldots, C_\lambda$ to the prover's hardware $\mathcal{H}_\mathsf{P}$.
3) $\mathcal{H}_\mathsf{P}$ applies these challenges to the SIMPL/PPUF, and determines the corresponding responses, which we call $\bar{R}_1, \ldots, \bar{R}_\lambda$. $\mathcal{H}_\mathsf{P}$ sends $\bar{R}_1, \ldots \bar{R}_\lambda$ to $\mathcal{H}_\mathsf{V}$.
4) $\mathcal{H}_\mathsf{V}$ measures the time period t^* that has passed between sending $C_1, \ldots, C_\lambda$ and receiving $\bar{R}_1, \ldots, \bar{R}_\lambda$.
5) $\mathcal{H}_\mathsf{V}$ then applies the following *decision rule*: *If* the responses $\bar{R}_1, \ldots, \bar{R}_\lambda$ were sent fast enough, i.e., if

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

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

$$\mathsf{Ver_S}(C_j, \bar{R}_j) = 1 \quad \text{for all } j = 1, \ldots, \lambda,$$

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

As in Scheme 10, some level of error tolerance can be achieved by appropriately relaxing the decision rule of $\mathcal{H}_\mathsf{V}$.

Analogously to the discussion following Scheme 10, Scheme 13 is secure against an adversary $\mathcal{A}$ as long as

$$t_{\mathsf{AdvPre}} \;\geq\; 2 \cdot t_{\mathsf{Lat}} + \lambda \cdot t_{\mathsf{Meas}}, \tag{2}$$

and if an $(\epsilon, t_{\mathsf{AdvPre}})$-SIMPL/PPUF with respect to $\mathcal{A}$ is used.

Pre-simulating the used λ CRPs prior to a protocol run $\mathsf{R_i}$ is possible for verifiers [4], [28], but has its limits: First of all, it requires knowledge that an identification with a specific SIMPL/PPUF is upcoming. Secondly, it unwantedly induces new secrets in the verifier's hardware, namely the precomputed and stored CRPs, making the scheme no longer secret-free.

D. Keys, Secrets, and Isolation Assumptions

Let us now return to our main topic, namely the *keys, secrets, and isolation assumptions* in Scheme 13:

- The *prover's hardware* $\mathcal{H}_\mathsf{P}$ is secret-free.
- Also the *verifier's hardware* $\mathcal{H}_\mathsf{V}$ is secret-free.
- *No* activatable isolation is assumed in $\mathcal{H}_\mathsf{P}$.

From a secret-related perspective, SIMPLs/PPUFs could thus be seen as a **direct extension** of standard public key cryptography: While the latter removes secrets from verifiers in identification, SIMPLs/PPUFs counterintuituvely remove them from both parties! Some thinking shows that a minimal requirement for any successful identification is that the verifier holds some possibly public, but at least authenticated information on the prover. SIMPLs/PPUFs are the first and only known primitive that enables remote identification in this minimal, secret-free setting.

VII. Secret-Free Security III: UNOs

A. Basic Idea and Definitions

Consider a small piece of paper ($0.01\,\mathrm{mm}^2$, say), which is scanned by some electron microscope, producing a high-resolution image $\overline{P}$ of its properties and structure. Even if $\overline{P}$ was known to adversaries, they 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. Our piece of paper hence must be considered secret-free and unclonable with respect to external measurement by the microscope. The above phenomenon seems like an interesting security property, which is not yet captured directly by one of our earlier notions. This motivates the introduction of the complementary concept of a Unique Object (UNO) [51].

Definition 14 (Standard Measurement Apparatuses)**.** *A* Standard Measurement Apparatus (SMA) *is a physical device* M*, which takes as input a physical object* O*, and outputs a binary string* $\mathsf{Prop_M(O)}$*, called the* properties of O (upon measurement with M)*, while leaving* O *unaltered.* M *shall be mass-producible, i.e., an arbitrary number of* specimens $\mathsf{M}', \mathsf{M}'', \ldots$ *shall be manufacturable that possesses the same input-output behavior.*

Given these preparations, we can now define unique objects.

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

CloneGame$(\mathsf{O}, \mathsf{M}, \mathcal{A}, \mathsf{Prop_M(O)}, \mathsf{InIn_O}, \mathsf{InIn_M}, \mathsf{M}')$:

$\mathcal{A}$ *is given* O*, a specimen* M' *of the mass-producible SMA* M, $\mathsf{Prop_M(O)}$, $\mathsf{InIn_O}$ *and* $\mathsf{InIn_M}$*. Within one year,* $\mathcal{A}$ *must output two physical objects* $\mathsf{O_1}$ *and* $\mathsf{O_2}$*, thereby only being limited in his actions by his technological capabilities and equipment.* $\mathcal{A}$ *wins the game if*

$$\mathsf{Prop_M(O)} = \mathsf{Prop_M(O_1)} = \mathsf{Prop_M(O_2)}.$$

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 $\mathsf{O_1}$ and $\mathsf{O_2}$ may be, but need not be equal to O. 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 $\mathsf{Prop_M(O)}$, $\mathsf{InIn_O}$ and $\mathsf{InIn_M}$ 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 order to break their security properties. In all earlier definitions, it was already sufficient for attacks if adversaries *could numerically output* a certain response correctly (and/or fast enough). This means that UNOs for the first time enforce real, physical cloning in security breaks. This can substantially enhance resilience against attacks, especially against adversaries with limited access to fabrication equipment, such as standard consumers.

Conceptually, the price to be paid is the presence of a trusted, external, mass-producible measurement apparatus, which can physically measures the UNO on site. Remote protocols between provers and verifiers are not intended in a UNO-context. Still, on-site measurements allow application to two societally extremely relevant problems: The unforgeable tagging of valuable items (see Section VII-C) and digital rights management (see Section VIII-D).

B. Implementation

From a theoretical perspective, the existence of UNOs is motivated by the fundamental 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 [40]. 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 Section VII-A).

Without using the term UNO (and mostly without developing an implementation-independent, conceptual theory behind it), a large number of researchers seems to have proposed and re-invented the basic idea of UNOs ad hoc and independently. The thread starts as early as in the late 1960s [34], continues in the 1980s [3], [22] and 1990s [59], [44], [45], [58], [66], [11], [24], just to surface again in the 2000s [10], [15], [68], [25], [8], [12], [60]. 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 a commodity, everyday structure. Different measurement methods have been proposed to extract its unique features, including lasers [8], [60] and ordinary scanners [12]. Other implementation suggestions include optical fibers [10], as well as radio wave [15] and magnetic [34], [66], [11] UNOs.

Again, we would like to propose two didactic, conceptual UNO-examples to readers, which they can hold in their minds throughout this section. The first one is the abovementioned, tiny paper surface (or any other suited disordered surface), when being scanned with a high-resolution microscope. The second is again related to Pappu et al.'s optical PUF, establishing a common thread within our discussion: Assume now that 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 [6], while positioning the 25 particles with the same accuracy in 3D is currently impossible [6]. Unclonability hence is upheld, *even if all information about the structure is known*, making it a UNO. At the same time, it is not complex enough to serve as Strong PUF, Complex PUF, or SIMPL/PPUF, differentiating these concepts from UNOs.

Finally, let us argue why XOR Arbiter PUFs by 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 serve as unique properties (UPs): Recall that an adversary know the target UPs in his attack, and that XOR Arbiter PUFs have a digital challenge-response interface. He 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. This illustrates again how secret-free UNOs with an external, analog measurement apparatus differ from secret-containing PUFs with integrated, digital challenge-response interfaces.

C. An Example Scheme and Its Properties

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 [64], with associated wider economic and social costs of \$737 to \$898 Billion [64], and employment losses of 2 to 2.6 million jobs [64]. Similar figures have been reported by the OECD [65] or Interpol [15].

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, it 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, it possesses the public verification key PK corresponding to SK, and an inbuilt 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 (see Table I).

Scheme 16: ON-SITE IDENTIFICATION OR UNFORGEABLE ITEM TAGS WITH UNOs [22], [15], [51]

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 mass-manufacturable SMA M.
2) Using his signing key SK, which is stored permanently in non-volatile 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 corre-

sponding to SK in non-volatile memory, and possesses its own specimen M'' of the mass-manufacturable 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

$$\mathsf{Prop}_{\mathsf{M}}(\mathsf{O}) = \mathsf{Prop}_{\mathsf{M}}(\overline{\mathsf{O}}), \tag{3}$$

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

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 privacy of verifiers, and maintains non-traceability of tagged items, such as banknotes. The *item of value* itself thereby acts as a store-and-forward channel, 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! Finally, the tag's digital signature can furthermore certify some additional, item-related information, such as biometric features of a passport owner, monetary value of banknotes, or consumer product data.

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 3. As before, some small error-tolerance and deviation may be allowed in this process, possibly using helper data [41].

D. Keys, Secrets, and General Perspective

Having motivated the existence of UNOs, and detailed their implementation and applications, let us now return to the *keys, secrets, and isolation assumptions* they induce in Scheme 16.

- The *item of value* and the *tag* are secret-free.
- The *testing device* is secret-free.
- The *manufacturer* contains classical keys.
- *No* activatable isolation is assumed in the item of value and the tag.

This establishes another identification-type scheme with secret-free features! One particular asset of its key- and secret-structure is that the two widespread components of the system – 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 protection, but may remain inexpensive. The *manufacturer*, on the other hand, does contain a secret, but may act like a trusted central authority, who creates his signatures from a remote, well-protected environment when being given $\mathsf{Prop}_{\mathsf{M}}(\mathsf{O})$.

Compared to SIMPLs/PPUFs, UNOs require an extra trusted, external measurement apparatus on site. But on the upside, they are far simpler to implement: Often non-electronic, cost-effective everyday media *(such as paper)* can be used. This promises particularly easy mass-market applicability.

VIII. Secret-Free Security IV: Extensions

The previous sections all focused on (remote or on-site) identification schemes, establishing a joint, common benchmark for our comparative analyses. This should not hide the significant number of other applications in which secret-free security can already be achieved today. This section will sketch several of these, providing pointers to the literature, and will also briefly touch upon limits and open questions.

A. Secret-Free Message Authentication

To start with, a full protocol for secret-free message authentication based on SIMPLs/PPUFs has been given in [46], [48]; it bears some similarities with Scheme 13. The protocol could theoretically be adapted to run with Complex PUFs, then merely achieving secret-free provers in the execution phase. Both protocols are interactive message authentication schemes: The verifier has to communicate with the prover online in order to test the authenticity of a message, and the entire protocol needs to be completed within a certain time frame. As earlier, this time frame is related to the adversarial prediction time t_{AdvPre} of the employed Complex PUF or SIMPL/PPUF (compare Schemes 10 and 13).

B. Secret-Free Tamper Detection

One of the earliest envisioned applications of PUFs has been tamper detection [41], [19]: To this end, optical PUFs (or other PUFs whose CRPs are sensitive to violations of their structural integrity) 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, capsule integrity and non-tamperedness of the inner hardware may be checked remotely. By using *Complex PUFs* or *SIMPLs/PPUFs* as capsules, this approach can be accomplished with a secret-free prover during the execution phase (Complex PUFs), or even with secret-free provers *and* verifiers (SIMPLs/PPUFs); compare Schemes 10 and 13.

C. Virtual Proofs of Reality and Secret-Free Sensing

Under the assumption that Complex PUFs or SIMPLs/PPUFs can be made reliably dependent in their responses on certain environmental variables, such as temperature, humidity, pressure, etc., their CRPs can *"prove"* the actual value of these variables to remotely located parties. To this end, protocols similar to Schemes 10 and 13 can be employed, in which each response is also a function of the desired environmental variable. This is the principle of so-called Virtual Proofs (VPs) of Reality [54]. VPs can be made partly *(or completely)* secret-free by using suitable, environment-dependent Complex PUFs *(or SIMPLs/PPUFs)*. Partly secret-free VPs of the destruction of a physical object, and of the spatial distance of two objects, have already been demonstrated in experiment by optical Complex PUFs [54], and so have key-free temperature sensors based on electrical PUFs [54]. Similar techniques could enable key-free or secret-free nuclear weapons inspections and disarmament verification [43].

D. Secret-Free Digital Rights Management

Storage media like CDs/DVDs also possess unique properties on a sub-digital, analog level [25], [68], even if they store exactly the same digital content. This allows their use as UNOs, and their application to the digital rights management problem. By a very small modification of Scheme 16, the digital content on CDs/DVDs can then be certified, again using digital signatures [25], [68]. Customary CD/DVD-readers can serve as widespread, inexpensive, but highly accurate measurement devices. Both CDs/DVDs and readers are *secret-free* in this approach, contrary to existing techniques [1].

E. Secret-Free Encryption and Digital Signatures?

Is secret-free encryption in our sense possible? Two basic obstacles exist. First of all, encryption requires long-term security against adversaries. This means that the employed Complex PUFs or SIMPLs/PPUFs would need extremely large t_{AdvPre}, while currently, their implementation can sometimes already be challenging for medium values of t_{AdvPre}. Secondly, the encryption hardware necessarily needs to receive the plaintext as input, which constitutes a secret in any encryption scheme. At least while this input is given to the hardware, it hence cannot be secret-free. Reduced forms of secret-freeness might still be achievable, for example encryption hardware that does not contain any keys or *digital* secrets. Similar arguments apply to the long-term security of secret-free digital signatures, turning both into an interesting future research activity.

IX. Summary and Future Work

In this conceptual and survey paper, we discussed the avoidance of keys and secrets in cryptography and security. Loosely speaking, a secret in our sense is any information that is present in arbitrary form in hardware, and whose disclosure to adversaries breaks security. Secret keys, in turn, are a special subclass of secrets, namely those that are present in volatile or non-volatile digital memory. Given the latest generation of physical security primitives, the design of not just key-free, but of completely secret-free hardware has become a realistic possibility. This concept paper and survey took up the stimulus resulting from this situation, trying to lay the foundations for future secret-free hardware. Among others, the latter would be innately immune against adversarial probing and key extraction attempts.

From a fundamental perspective, the two basic phenomena that enable the realization of secret-free hardware are the *fabrication complexity* and *simulation complexity* of physical systems: Even if the entire internal structure of a given physical hardware system *(i.e., all its internal "secrets")* are known, it may still be infeasible to clone it, and/or to emulate its input-output behavior in real-time. The role of these two assumptions is comparable to the infamous computational complexity assumptions in classical cryptography. But they enable a qualitatively new security feature: Namely removing all secrets from hardware.

Motivated by the described situation, this paper is to our knowledge the first to announce, survey, and categorize the fields of key-free and, more importantly, secret-free security. We defined the core concepts of a secret, secret-free hardware, isolation assumptions in hardware, and a taxonomy of keys and secrets. Applying our novel framework, we classified a large number of existing schemes with respect to their exact keys, secrets, and isolation assumptions. Table I on page 17 shows our findings: It illustrates that standard, silicon PUFs can achieve initial forms of key-free, but no secret-free security yet. But three other physical primitives known as Complex PUFs, SIMPLs/PPUFs, and UNOs actually can!

We unfolded these three primitives in detail, including their implementations and protocol uses. Within this process, we developed the first coherent definitional framework for said three primitives, introducing the concept of *internal information* of hardware and physical objects. Providing *any* internal information of *any* involved hardware to attackers, while still requiring that security should be upheld, turned out to be a generic technique for defining secret-free security. In addition, we suggested the new concept of a Complex PUF, which acts as gentle link between the key-free world of PUFs and the secret-free world beyond. Finally, Section VIII showed that the reach of secret-free methods also encompasses secret-free message authentication, tamper detection, sensing, and digital rights management. This demonstrates two aspects: Firstly, that secret-free security has become a viable possibility already; secondly, that it is not limited to pure identification applications.

A host of research opportunities, perhaps even some research program, evolves from the presented material. On the conceptual and theory side, formulating and proving the security of secret-free schemes in advanced models, such as the universal composition framework, seems very interesting. Investigating necessary conditions for secret-free security, formally showing statements like *"secret-free provers in secure remote identification imply the existence of Complex PUFs"*, would be intriguing, too. So would be the application of our generic framework of keys, secrets, and isolation assumptions to other security methods and paradigms. Finally, identifying further secret-free primitives would be fascinating — or, alternatively, proving that none exist!

Secondly, on the practical side, the optimized design, implementation, and commercial deployment of the presented concepts in hardware represents a major and long-term endeavour. Just to name three examples: The development of CMOS-compatible Complex PUFs or SIMPLs/PPUFs; of secret-free sensors for various physical variables beyond the known VPs of temperature [54]; or of UNOs/unforgeable item tags that can be measured at high security levels by consumer smart phones; all represent seminal, but nevertheless realistic goals. We hope that the work in this paper can inspire and foster such future research activities.

References

[1] R. J. Anderson: *Security Engineering: A guide to building dependable distributed systems*. Wiley, 2010.

[2] F. Armknecht, R. Maes, A.R. Sadeghi, F.X. Standaert, C. Wachsmann: *A formalization of the security features of physical functions.* IEEE Symposium on Security & Privacy, 2011.

[3] D.W. Bauder: *An anti-counterfeiting concept for currency systems*. Sandia Na- tional Labs, Albuquerque, NM, Tech. Rep. PTK-11990, 1983.

[4] N. Beckmann, M. Potkonjak: *Hardware-based public-key cryptography with pub- lic physically unclonable functions.* Information Hiding 2009, pp. 206-220, 2009.

[5] C.H. Bennett, G. Brassard: *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] B. van den Broek, B. Ashcroft, T. H. Oosterkamp, J. van Noort: *Parallel nanometric 3D tracking of intracellular gold nanorods using multifocal two-photon microscopy*. Nano Letters, 13(3), 980-986, 2013.

[7] C. Brzuska, M. Fischlin, H. Schröder, S. Katzenbeisser: *Physical Unclonable Functions in the Universal Composition Framework.* CRYPTO 2011.

[8] J. Buchanan, R. Cowburn, A. Jausovec, D. Petit, P. Seem, G. Xiong, D. Atkinson, K. Fenton, D. Allwood, and M. Bryan: *Forgery: Fingerprinting documents and packaging.* Nature, vol. 436, no. 7050, p. 475, 2005.

[9] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, U. Rührmair: *The Bistable Ring PUF: A New Architecture for Strong Physical Unclonable Functions*. HOST 2011.

[10] Y. Chen, M. K. Mihcak, D. Kirovski: *Certifying authenticity via fiber-infused paper*. SIGecom Exchanges 5(3): 29-37 (2005)

[11] M.C. Chu, L.L. Cheng, L.M. Cheng: *A novel magnetic card protection system.* European Convention on Security and Detection, pp. 207-211, 1995.

[12] W. Clarkson, T. Weyrich, A. Finkelstein, N. Heninger, J. Halderman, E. Felten: *Fingerprinting blank paper using commodity scanners.* IEEE Symposium on Security and Privacy (Oakland'09), pp. 301-314, 2009.

[13] C.T. Clelland, V. Risca, C. Bancroft: *Hiding messages in DNA microdots.* Nature 399.6736, pp. 533-534, 1999.

[14] G. Csaba, X. Ju, Z. Ma, Q. Chen, W. Porod, J. Schmidhuber, U. Schlichtmann, P. Lugli, U. Rührmair: *Application of Mismatched Cellular Nonlinear Networks for Physical Cryptography*. IEEE Workshop on Cellular Nanoscale Networks and Their Applications (CNNA), pp. 1-6, 2010.

[15] G. DeJean, D. Kirovski: *RF-DNA: Radio-Frequency Certificates of Authenticity*. CHES 2007: 346-363

[16] S. Deyati, B.J. Muldrey, A.D. Singh, A. Chatterjee: *Challenge engineering and design of analog push pull amplifier based physically unclonable function for hardware security*. IEEE Asian Test Symposium (ATS), pp. 127-132, 2015.

[17] R.P. Feynman: *Simulating physics with computers.* International Journal of Theoretical Physics 21.6-7, pp. 467-488, 1982.

[18] S. L. Garfinkel, A. Juels, R. Pappu. *RFID privacy: An overview of problems and proposed solutions.* IEEE Security & Privacy 3.3: 34-43, 2005.

[19] B. Gassend, *Physical Random Functions*, MSc Thesis, MIT, 2003.

[20] B. Gassend, D. E. Clarke, M. van Dijk, S. Devadas: *Silicon physical random functions*. ACM Conference on Computer and Communications Security 2002, pp. 148-160, 2002

[21] O. Goldreich: *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2009.

[22] R.N. Goldman: *Non-counterfeitable document system*. US-Patent 4,423,415. Publication date: 1983. Priority date: 1980. See https://patents.google.com/patent/US4423415A

[23] J. Guajardo, S. S. Kumar, G. J. Schrijen, P. Tuyls: *FPGA Intrinsic PUFs and Their Use for IP Protection*. CHES 2007: 63-80

[24] T. Haist, H.J. Tiziani: *Optical detection of random features for high security applications*. Optics Communication 147, pp. 173-179, 1998.

[25] G. Hammouri, A. Dana, B. Sunar: *CDs have fingerprints too.* CHES 2009: 348-362.

[26] C. Helfmeier, C. Boit, D. Nedospasov, J.-P. Seifert: *Cloning Physically Unclonable Functions*. HOST 2013: 1-6

[27] C. Herder, M.D. Yu, F. Koushanfar, S. Devadas: *Physical unclonable functions and applications: A tutorial.* Proceedings of the IEEE 102(8), pp. 1126-1141, 2014.

[28] C.H. Herder: *Towards security without secrets.* PhD Thesis, Massachusetts Institute of Technology (MIT), 2016.

[29] D. E. Holcomb, W. P. Burleson, K. Fu: *Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers*. IEEE Trans. Computers 58(9): 1198-1210, 2009.

[30] C. Jaeger, M. Algasinger, U. Rührmair, G. Csaba, M. Stutzmann: *Random pn-junctions for physical cryptography*. Applied Physics Letters 96(17), 172103, 2010.

[31] A. Kent: *Unconditionally secure bit commitment by transmitting measurement outcomes.* Physical review letters 109.13, 130501, 2012.

[32] A. Kerckhoffs: *La cryptographie militaire.* Journal des sciences militaires, Vol. IX, pp. 5-38, 1883.

[33] A. Leier, C. Richter, W. Banzhaf, H. Rauhe: *Cryptography with DNA binary strands*. Biosystems 57(1), pp. 13-22, 2000.

[34] G. Lindstrom, G. Schullstrom: *Verifiable identification document.* US-Patent 3,636,318. Publication date: 1972. Priority date: 1968. See https://patents.google.com/patent/US3636318A

[35] K. Lofstrom, W.R. Daasch, D. Taylor: *IC identification circuit using device mismatch.* ISSCC 2000, pp. 372-373, 2000.

[36] R. Maes, I. Verbauwhede: *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.

[37] M. Majzoobi, F. Koushanfar: *Time-bounded authentication of FPGAs.* IEEE Transactions on Information Forensics and Security 6.3, p. 1123-1135, 2011.

[38] U.M. Maurer: *Secret key agreement by public discussion from common information.* IEEE transactions on information theory 39.3, pp. 733-742, 1993.

[39] D. Nedospasov, J.-P. Seifert, C. Helfmeier, C. Boit: *Invasive PUF Analysis*. FDTC 2013: 30-38

[40] R. Pappu: *Physical One-Way Functions*. PhD Thesis, Massachusetts Institute of Technology, 2001.

[41] R. Pappu, B. Recht, J. Taylor, N. Gershenfeld: *Physical One-Way Functions*, Science, vol. 297, pp. 2026-2030, 20 September 2002.

[42] M. Potkonjak, V. Goudar: *Public physical unclonable functions.* Proceedings of the IEEE 102.8 (2014): 1142-1156.

[43] S. Philippe, M. Kütt, M. McKeown, U. Rührmair, A. Glaser: *The Application of Virtual Proofs of Reality to Nuclear Safeguards and Arms Control Verification*. In 57th Annual INMM Meeting, 2016.

[44] R.L. van Renesse: *3DAS: a 3-dimensional-structure authentication system.* European Convention on Security and Detection, pp. 45-49, 1995.

[45] R.L. van Renesse: *Optical document security.* Boston: Artech House, 1998.

[46] U. Rührmair: *SIMPL Systems: On a Public Key Variant of Physical Unclonable Functions.* IACR Cryptology ePrint Archive, Report 2009/255, 2009.

[47] U. Rührmair: *SIMPL Systems, Or: Can we build cryptographic hardware without secret key information?* SOFSEM 2011. Lecture Notes in Computer Science, Vol. 6543, Springer, 2011.

[48] U. Rührmair: *SIMPL Systems as a Keyless Cryptographic and Security Primitive*. In: D. Naccache (Editor), Cryptography and Security: From Theory to Applications. Lecture Notes in Computer Science, Vol. 6805, Springer, 2012.

[49] U. Rührmair: *Oblivious Transfer based on Physical Unclonable Functions.* TRUST 2010.

[50] U. Rührmair: *Physical Turing Machines and the Formalization of Physical Cryptography*. Cryptology ePrint Archive, Report 2011/188, 2011.

[51] U. Rührmair, S. Devadas, F. Koushanfar: *Security based on Physical Unclonability and Disorder.* In M. Tehranipoor and C. Wang (Editors): Introduction to Hardware Security and Trust. Springer, 2011

[52] U. Rührmair, M. van Dijk: *Practical Security Analysis of PUF-Based Two-Player Protocols.* pp. 251-267, CHES 2012.

[53] U. Rührmair, D. E. Holcomb: *PUFs at a Glance*. DATE 2014.

[54] U. Rührmair, J.L. Martinez-Hurtado, X. Xu, C. Kraeh, C. Hilgers, D. Kononchuk, J.J. Finley, and W.P. Burleson: *Virtual Proofs of Reality and their Physical Implementation*. IEEE Symposium on Security and Privacy, pp. 70-85, 2015.

[55] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, J. Schmidhuber: *Modeling Attacks on Physical Unclonable Functions*. ACM Conference on Computer and Communications Security, 2010.

[56] U. Rührmair, J. Sölter, F. Sehnke: *On the Foundations of Physical Unclonable Functions.* IACR Cryptology ePrint Archive, 2009.

[57] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, M. Majzoobi, F. Koushanfar, W. P. Burleson: *Efficient Power and Timing Side Channels for Physical Unclonable Functions*. CHES 2014: 476-492

[58] J.R. Smith, A.V. Sutherland: *Microstructure based indicia.* Proceedings of the Second Workshop on Automatic Identification Advanced Technologies, 1999.

[59] G.J. Simmmons: *Identification of data, devices, documents and individuals.* Annual International Carnahan Conference on Security Technology, pp. 197-218, 1991.

[60] A. Sharma, L. Subramanian, E. A. Brewer: *PaperSpeckle: microscopic fingerprinting of paper.* Proceedings of the 18th ACM conference on Computer and communications security (CCS'18), 2011.

[61] M. Stutzmann, G. Csaba, P. Lugli, J.J. Finley, C. Jirauschek, C. Jaeger, U. Rührmair: *Towards Electrical, Integrated Implementations of SIMPL Systems*. European Patent Application EP2230794 A3. Priority date: March 16, 2009. See https://patents.google.com/patent/EP2230794A3

[62] G.E. Suh, S. Devadas: *Physical unclonable functions for device authentication and secret key generation.* Design Automation Conference, pp. 9-14, 2007.

[63] S. Tajik, E. Dietz, S. Frohmann, J.-P. Seifert, D. Nedospasov, C. Helfmeier, C. Boit, H. Dittrich: *Physical Characterization of Arbiter PUFs*. CHES 2014: 493-509

[64] *The Economic Impacts of Counterfeiting and Piracy — Executive Summary*. International Chamber of Commerce BASCAP and INTA Frontier Reports, 2017. Download from https://iccwbo.org/publication/economic-impacts- counterfeiting-piracy-report-prepared-bascap-inta/

[65] *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

[66] A.W. Vaidya: *Keeping card data secure at low cost.* European Convention on Security and Detection, pp. 212-215, 1995.

[67] A. Vijayakumar, S. Kundu: *A novel modeling attack resistant PUF design based on non-linear voltage transfer characteristics*. DATE 2015: 653-658

[68] D. Vijaywargi, D. Lewis, D. Kirovski: *Optical DNA*. Financial Cryptography 2009: 222-229

[69] A.C.C. Yao: *Classical physics and the Church–Turing Thesis.* Journal of the ACM (JACM) 50.1, pp. 100-105, 2005.

Appendix

A. Table of Our Main Results

Table I on page 17 overviews at one glance some of the main findings of this paper. It illustrates the stepwise transition from standard schemes, that contain classical keys and employ activatable isolation, to the most advanced physical approaches, which are completely secret-free and do not require activatable isolation. Many of the described features provable cannot be achieved in a purely mathematical, Turing machine based setting. Further explanations are given in the caption of the table.

B. Idealized Weak PUFs in Symmetric Identification

To which degree are the keys and secrets of Scheme 6 caused by the considered implementation example of SRAM PUFs, and to which extent are they inherent to the Weak PUF concept itself? In order to address this question, let us define an abstract, standardized version of plain, silicon Weak PUFs with optimal properties (an *Idealized Weak PUF*, for short) as follows:

Definition 17 (Idealized Weak PUFs)**.** *An Idealized Weak PUF* P *is assumed to be a "black box" with a digital response interface. When triggered with some fixed stimulus* S*, it converts its internal manufacturing variations* $\mathsf{ManV_P}$ *into a single, fixed, multi-bit digital response* $\overline{R}$*, which is being output over this interface. We suppose that* $\overline{R}$ *has perfect entropy, is perfectly stable and noise-free, and is a sole function of* $\mathsf{ManV_P}$*, i.e.*:

$$\overline{R} = F(\mathsf{ManV_P}).$$

Furthermore, F *is assumed to be efficiently computable and publicly known, also to adversaries. Apart from its internal manufacturing variations and randomness, an Idealized Weak PUF is supposed to contain no further secrets.*

Let us now assume that such an *Idealized Weak PUF* was employed in Scheme 6 instead of an SRAM PUF. Due to its perfect entropy and stability, it can be directly used as secret key K, making error correction superfluous. But otherwise, Scheme 6 runs exactly the same, also with Idealized Weak PUFs, inducing the following *keys, secrets, and isolation assumptions*:

(i) The *prover's hardware* $\mathcal{H}_\mathsf{P}$ now is not just classically key-free (as in the case of SRAM PUFs), but even key-free. But if $\mathcal{H}_\mathsf{P}$ is a fully digital system, it will contain various transient digital secrets: Firstly, the signals that communicate $\overline{R} = \mathsf{K}$ over the digital response interface of the Idealized Weak PUF. Secondly, the digital secrets that transfer K to the circuitry within $\mathcal{H}_\mathsf{P}$ that computes $\mathsf{PRF_K(N_i)}$. Thirdly, there will usually be further transient digital secrets in the circuitry computing $\mathsf{PRF_K(N_i)}$ itself, their exact nature depending on the choice of PRF and its implementation. Finally, $\mathcal{H}_\mathsf{P}$ also contains physical secrets, namely the physical manufacturing variations $\mathsf{ManV_P}$ from which $\overline{R}$ can be computed. (This assumes that F can be efficiently computed, as stipulated in Definition 17).

(ii) The *verifier's hardware* $\mathcal{H}_{\mathsf{V}}$ contains a non-volatile digital secret, or classical key, namely K. $\mathcal{H}_{\mathsf{V}}$ is hence not even classically key-free.

(iii) The scheme again assumes activatable isolation in $\mathcal{H}_{\mathsf{P}}$.

The only notable enhancement from replacing SRAM PUFs by Idealized Weak PUFs therefore concerns the prover's hardware: It is now key-free instead of merely classically key-free. All other keys, secrets, and isolation assumptions remain the same as in the case of SRAM PUFs (compare Section IV-A). This suggests two implications: Firstly, current silicon practice in Weak PUF implementation is relatively close to optimal, at least from the conceptual perspective of keys, secrets, and isolation assumptions. Secondly, no completely secret-free hardware can be achieved by plain, silicon Weak PUFs, not even by idealized ones. Yet more powerful concepts are required to this end, as discussed in Sections V to VIII.

Finally, let us comment in passing that Idealized Weak PUFs are not merely theoretical or far-fetched. Certain existing implementations, that are not based on volatile memory cells, could be regarded as first steps towards their realization. This includes, among others, transistor-based [35] and diode-based [30] Weak PUFs, to which we refer interested readers for further reading.

C. Idealized Strong PUFs in CRP-based Identification

Similar to Appendix B, we ask in this section: Are the keys and secrets in Scheme 7 mainly caused by the considered example implementation of XOR Arbiter PUFs? Or are they inherent to the Strong PUF concept itself? Once again, defining an *idealized* version of plain, silicon Strong PUFs turns out helpful.

Definition 18 (Idealized Strong PUFs)**.** *An* Idealized Strong PUF P *is assumed to be a "black box" with a digital challenge-response interface. Whenever a digital challenge C_i is applied via the interface, the Idealized Strong PUF generates a perfectly stable, noise-free digital response R_i, which solely is a function of the internal manufacturing variations* $\mathsf{ManV_P}$ *and the challenge C_i:*

$$R_i = F(\mathsf{ManV_P}, C_i).$$

Furthermore, F is assumed to be efficiently computable and publicly known, also to adversaries. The challenge-response pairs of the Idealized Strong PUF shall constitute a pseudo-random function that is computationally secure against existential forgery [1], [21]. Apart from its internal manufacturing variations, the Idealized Strong PUF is supposed to contain no further secrets.

Let us now suppose that such an *Idealized Strong PUF* is employed in the CRP-based identification of Scheme 7 instead of an XOR Arbiter PUF. The scheme then essentially remains the same, even though the implicit error correction no longer is necessary: Recall that Idealized Strong PUFs are perfectly stable by Definition 18.

If carried out with Idealized Strong PUFs instead of XOR Arbiter PUFs, Scheme 7 overall induces the following *keys, secrets and isolation assumptions*:

(i) The *prover's hardware* $\mathcal{H}_{\mathsf{P}}$ is key-free throughout the entire scheme, and also free of digital secrets in the execution phase. In the set-up phase, $\mathcal{H}_{\mathsf{P}}$ will contain transient digital secrets (namely the digital responses R_i that are being output by the Idealized Strong PUF's digital interface: Knowing them will allow adversaries impersonation attacks in any later run). Furthermore, $\mathcal{H}_{\mathsf{P}}$ contains physical secrets, namely the manufacturing variations $\mathsf{ManV_P}$ (we stress that this assumes that F is efficiently computable, as stipulated in Definition 18).

(ii) The *verifier's hardware* $\mathcal{H}_{\mathsf{V}}$ contains a non-volatile digital secret, or classical key, namely the CRP-List. $\mathcal{H}_{\mathsf{V}}$ is hence not even classically key-free.

(iii) *No* activatable isolation in $\mathcal{H}_{\mathsf{P}}$ is assumed.

Our analysis reveals the significant potential gains from replacing XOR Arbiter PUFs by Idealized Strong PUFs: Classical and volatile keys are avoided in $\mathcal{H}_{\mathsf{P}}$, and even digital secrets are evaded during the execution phase. Given these advancements, our findings suggest that current silicon Strong PUF designs are not optimal from a perspective of keys and secrets yet. This motivates future research on novel Strong PUF designs. At the same time, however, our discussion also shows that even Idealized Strong PUFs cannot achieve *secret-free* security. This once more confirms that new, more powerful approaches are required to this end, as the ones detailed in Sections V to VIII.

Finally, our comparison of *Idealized* Weak and Strong PUFs in Appendices B and C illustrates some of the theoretical advantages of the Strong PUF over the Weak PUF concept: Strong PUFs, at least conceptually, possess a demonstrably better potential to avoid digital secrets and isolation assumptions. This fact is also subsumed at a glance in Table I, which illustrates the steady transition from key-free to secret-free security.

Security Scheme	**Primitive/Method**	**Keys & Secrets at Verifier** *(or Testing Device, resp.)*	**Keys & Secrets at Prover** *(or Item and Tag, resp.)*	**Isolation Assumptions at Prover** *(or Item and Tag, resp.)*
Remote Identification *(via standard approach* [1]*)*	Symmetric Cryptography	Classical Keys	Classical Keys	Activatable Isolation
Remote Identification *(via standard approach* [1]*)*	Asymmetric Cryptography	Public Keys. **Secret-Free**	Classical Keys	Activatable Isolation
Remote Identification *(via Scheme 6)*	SRAM PUFs and Symmetric Cryptography	Classical Keys	Volatile Keys, Transient Digital Secrets, Non-Volatile Physical Secrets. **Classically Key-Free**	Activatable Isolation
Remote Identification *(via Scheme 7)*	XOR Arbiter PUFs	Classical Keys	Volatile Keys, Transient Digital Secrets, Non-Volatile Physical Secrets. **Classically Key-Free**	No Activatable Isolation
Virtual Proof of Temperature *(via Protocol 1 of* [54]*)*	XOR Bistable Ring PUFs	Classical Keys	Volatile Keys, Transient Digital Secrets, Non-Volatile Physical Secrets. **Classically Key-Free**	No Activatable Isolation
Remote Identification *(via Scheme 6 and App. B)*	Idealized Weak PUFs and Symmetric Cryptography	Classical Keys	Transient Digital Secrets, Non-Volatile Physical Secrets. **Key-Free**	Activatable Isolation
Remote Identification *(via Scheme 7 and App. C)*	Idealized Strong PUFs	Classical Keys	Transient Digital Secrets *(merely in Set-Up Phase)*, Non-Volatile Physical Secrets. **Key-Free**	No Activatable Isolation
Remote Identification *(via Scheme 10)*	Optical Complex PUFs	Classical Keys	Volatile Physical Secrets *(merely in Set-Up Phase).* **Secret-Free** *(in all of Execution Phase)*	No Activatable Isolation
Virtual Proof of Distance *(via Protocol 2 of* [54]*)*	Optical Complex PUFs	Classical Keys	Volatile Physical Secrets *(merely in Set-Up Phase).* **Secret-Free** *(in all of Execution Phase)*	No Activatable Isolation
Remote Identification *(via Scheme 13)*	SIMPLs/ PPUFs	**Secret-Free**	**Secret-Free**	No Activatable Isolation
On-Site Identification/ Unforgeable Item Tagging *(via Scheme 16)*	UNOs	**Secret-Free**	**Secret-Free**	No Activatable Isolation

TABLE I
Keys, secrets, and isolation assumptions in provers and verifiers when various physical primitives are used for remote identification *(and item tagging)*. The table emphasizes the steady transition from classically key-free over key-free to finally secret-free security. Any secret-free hardware by definition is key-free, and any key-free hardware by definition is classically key-free, but not vice versa *(Definitions 3 and 4)*, making *secret-free* the strongest feature. For completion, the employed isolation assumptions are listed in the *rightmost column*. Activatable isolation is a common, but rather strong assumption, whence schemes without it are preferable. Finally, we emphasize once more that that the presence of secrets in vulnerable hardware is not just an academic observation: *All* keys and secrets of the table have been extracted by different, dedicated attacks in the past *in practice* [1], [55], [39], [26], [63], [57]. This strongly motivates future research on secret-free hardware.