SIMPL Systems as a Keyless Cryptographic and Security Primitive

Ulrich Rührmair

***Abstract*—We discuss a recent cryptographic primitive termed *SIMPL system*. Like Physical Unclonable Functions (PUFs), SIMPL systems are disordered, unclonable physical systems with many possible inputs and a complex input-output behavior. Contrary to PUFs, however, each SIMPL system comes with a publicly known, individual numeric description that allows its slow simulation and output prediction. While everyone can determine a SIMPL system's output slowly by simulation, only its actual holder can determine the output fast by physical measurement. This added functionality allows new public key like protocols and applications.**

But SIMPLs have a second, perhaps more striking advantage: No secret information is, or needs to be, contained in SIMPL systems in order to enable cryptographic security. Neither in the form of a standard digital key, nor as secret information hidden in the random, analog features of some hardware, as it is the case for PUFs. The security of SIMPL systems instead rests on (i) an assumption regarding their physical unclonability, and (ii) a computational assumption on the complexity of simulating their output. This provides SIMPL systems with a natural immunity against any key extraction attacks, including malware, side channel, invasive, and modeling attempts.

In this manuscript, we give a comprehensive discussion of SIMPLs as a cryptographic and security primitive. Special emphasis is placed on the different cryptographic protocols that are enabled by this new tool.

***Index Terms*—SIMPL Systems, Public Key Cryptography, Physical Unclonable Functions, Hardware Security.**

I. Introduction

A. Background and Motivation

Electronic communication and security devices are pervasive in our life. Just to name two examples, around five billion mobile phones are currently in use worldwide [1], [2], and the world market of smart cards has an estimated volume of over three billion pieces per year [3], [4]. Their widespread use makes such devices both a well-accessible and a worthwhile target for adversaries. Many security attacks thereby are not targeted against the employed cryptographic primitives themselves, some of which have proven attack-resilient over surprisingly long time spans. Instead, they try to extract the employed secret keys by physical or software methods. Such key-extracting strategies are not just a theoretical concern, but have been demonstrated several times in widespread, commercial systems [5], [6], [7]. This drives the quest for new mechanisms that protect — or better still: avoid! — the presence of secret keys in vulnerable hardware system.

Ulrich Rührmair is with the Department of Electrical Engineering and Information Technology, Technische Universität München, 80333 München, Germany. E-mail: ruehrmair@in.tum.de.

B. Physical Unclonable Functions (PUFs)

The security primitive of a Physical Unclonable Function (PUF) [8], [9], [10], [11] was introduced, at least in part, in order to address some of the above problems. A PUF is a (partly) disordered physical system S that can be challenged with so-called external stimuli or challenges C_i, upon which it reacts with corresponding responses termed R_{C_i}. Contrary to standard digital systems, a PUF's responses shall depend on the nanoscale structural disorder present in the PUF. This disorder cannot be cloned or reproduced exactly, not even by its original manufacturer, and is unique to each PUF. Assuming the stability of the PUFs responses, any PUF S hence implements an individual function F_S that maps challenges C_i to responses R_{C_i}. Due to its complex and disordered structure, a PUF can avoid some of the shortcomings associated with digital keys. For example, it is usually harder to read out, predict, or derive its responses than to obtain the values of digital keys that are stored in non-volatile memory. This fact has been exploited for various PUF-based security protocols [8], [9], [15], [28].

One prominent example are PUF-based identification schemes [8], [9], [10]. They are usually run between a central authority (CA) and a hardware carrying a (unique) PUF S. One assumes that the CA had earlier access to S, and could establish a large, secret list of challenge-response-pairs (CRPs) of S. Whenever the hardware wants to identify itself to the CA at some later point in time, the CA selects some CRPs at random from this list, and sends the challenges contained in these CRPs to the hardware. The hardware applies these challenges to S, and sends the obtained responses to the CA. If these responses match the pre-recorded responses in the CRP-list, the CA believes the identity of the hardware.

C. Private Key like Functionality of PUFs

The described protocol has several well-known advantages [8], [9]. However, one potential downside is that it presumes a previously shared piece of secret numerical information (i.e., the CRP-list). This information needs to be established in a secure set-up phase between the CA and the hardware, and must constantly be kept secret. Furthermore, the CRP-list uses up over time, since no single CRP should be used more than once in the identification process, and hence must be large. In these aspects, PUFs are resemblant of classical private key systems.

D. Secret Information in PUFs

Another noteworthy point is that PUFs in general do not obviate the presence of secret information within cryptographic

hardware. The secret information is no longer stored in digital form in two-level systems, such as digital secret keys stored in non-volatile memory cells. But there is still some sort of secret information present in most PUFs, whose disclosure breaks the security of the system. Let us name two examples: In the case of SRAM PUFs the information that needs to be kept secret is the state of the SRAM cells after power up, or the tiny manufacturing variations of the SRAM cells that determine their state after power up [30]. Once this information is known to an adversary, he can numerically derive the same key as the cryptographic hardware embedding the SRAM PUF, and break the system. In the case of Arbiter PUFs, the secret information are the internal runtime delays in the circuit stages [11]. If this information is known, the adversary can numerically simulate the behavior of the PUF output by an additive, linear model, again breaking its security [31].

In other words, the architectures of most current PUFs "hide" or "obfuscate" secret, security-relevant information very well in analog characteristics of integrated circuits. But at the same time, they do not avoid the need for secret information in hardware systems in principle; they just store it in a different form.

E. Our Contributions

Our main contribution in this paper is the introduction and discussion of so-called SIMPL systems as a new security primitive, where the acronym SIMPL stands for SIMulation Possible, but Laborious. We present the first formal specification of SIMPL systems, and show that they can implement a multitude of communication protocols, including identification, message authentication, coin flipping, bit commitment, and zero-knowledge proofs. We analyze scenarios in which these protocols can be applied, including secure communication in networks, item tagging and digital rights management. Furthermore, we survey existing hardware implementation candidates. Emphasis is placed on the broad cryptographic usability of SIMPLs, and on their potential to construct security hardware without secret key information.

F. Related Work

The current paper is an extended version of [16] and [20]. Since [16], several follow-up papers of our group have focused on the implementation of SIMPLs by electrical circuits [17], [18], [19], [21] and optical structures [20]. We emphasize that around the same time as [16], a comparable concept has been described completely independently in [24] under the name of a Public PUF (PPUF), and has been applied for key exchange purposes. It builds on a ideas and hardware architectures discussed already in [25]. Another closely related, but later idea is the concept of time-bounded authentication (TBA) [26], which has been suggested for identification schemes on FPGAs.

G. Organization of this Paper

The rest of this manuscript is organized as follows: In Section II, we give a semi-formal specification of SIMPL systems, and discuss their properties. Sections III to V discusses protocols that can be realized on the basis of SIMPL systems and PPUFs, starting with identification and message authentication (Sec. III), two-player protocols (Sec. IV), and key exchange (Sec. V). Section VI treats applications of SIMPL systems, and Section VII surveys the existing implementation candidates. We conclude the paper in Section VIII.

II. Specification and Properties of SIMPL Systems

A. Informal Description

We start this section by an informal description of the notion of a SIMPL system [1]. A physical system S is called a *SIMPL system* (or just a *SIMPL*) if the following holds:

1) S is a partly disordered physical system. It can be stimulated with challenges C_i, upon which it reacts with corresponding responses R_{C_i}. The responses are a function of the specific disorder present in S and of the applied challenge C_i.
2) The responses are assumed to be sufficiently stable to regard the behavior of S as a function F_S that maps challenges C_i to responses R_{C_i}. The pairs of the form (C_i, R_{C_i}) are often called the challenge-response pairs or CRPs of S.
3) It is possible (at least for the original manufacturer of S) to derive an individual numeric description $D(S)$ of S and an algorithm Sim. By use of $D(S)$ and Sim, everyone can simulate the correct responses R_{C_i} of S to any challenges C_i, or can at least verify a purported response R_{C_i} to a challenge C_i for correctness.
4) Any numeric simulation and any physical emulation that can predict the responses of S is noticeably slower than the real-time behavior of S. This must hold for simulation via Sim and $D(S)$, but must also apply to any adversarial algorithms and physical emulators. It must be upheld if the adversary has knowledge of $D(S)$, Sim, of all internal characteristics and disorder of S, and had earlier access to S.
5) It is difficult to physically clone S, i.e., to produce a "copy" S' which generates the same responses as S with comparable speed. Again, this must hold even for an adversary who knows $D(S)$, Sim, the internal characteristics and disorder of S, and who had earlier access to S.

Under these circumstances, a SIMPL system S computes the publicly known, publicly computable function F_S *faster* than anything or anyone else. In particular, the holder of S can determine the function value $F_S(C_i) = R_{C_i}$ for a randomly chosen challenge C_i faster than any adversary. This feature lies at the heart of all SIMPL-based security protocols.

Interestingly, the concept of a SIMPL is related to some well-known work of Feynman, who investigated the Turing-simulatability of physical systems in [32]. He conjectured that (i) all physical systems can, in principle, be simulated by Turing machines, but that (ii) such simulation cannot always

[1] As mentioned in Section I-E, the acronym SIMPL stands for SIMulation Possible, but Laborious

be carried out in real time and will create a computational overhead [32]. SIMPL systems can be seen as a special application of these ideas in cryptography and security, combining them with the recent concept of physical unclonability.

B. Semi-Formal Security Specification

The above properties can be coined into a semi-formal security specification of SIMPL systems. Its style follows the specifications presented in [27], [28]. The specification describes the security of SIMPL systems as a "game" with the adversary, thereby introducing a relatively precise, parametric adversarial model.

Specification 1 (($t_{max}, c, t_C, t_{Ph}, q, \epsilon$)-SIMPL SYSTEMS.)**.** *Let S be a physical system mapping challenges C_i to responses R_{C_i}, with $\mathbf{C}$ denoting the finite set of all possible challenges. Let $c > 1$ be a constant, and let furthermore t_{max} be the maximum time (over all challenges $C_i \in \mathbf{C}$) which it takes until the system S has generated the response R_{C_i} to the challenge C_i.*

S is called a ($t_{max}, c, t_C, t_{Ph}, q, \epsilon$)-SIMPL SYSTEM *if there is a string $D(S)$, called the description of S, and a computer algorithm* Sim *such that the following conditions are met:*

1) *For all challenges $C_i \in \mathbf{C}$, the algorithm* Sim *on input $\big(C_i, D(S)\big)$ outputs R_{C_i} in feasible time.*
2) *For all binary strings X of length q, any cryptographic adversaries Eve will* SUCCEED *in the following* **security experiment** *with a probability of at most ϵ:*
   a) *Eve is given the string X, the numerical description $D(S)$ and the code of the algorithm* Sim *for a time period of length t_C.*
   b) *Within the above time period t_C, Eve is furthermore given physical access to the system S at adaptively chosen time points, and for time periods of adaptively chosen lengths. The only restriction is that her access times must add up to a total of at most t_{Ph}.*
   c) *After the time period t_C has expired, Eve is presented with a challenge C^* that was chosen uniformly at random from the set $\mathbf{C}$, and is asked to output a value V_{Eve}.*

We thereby say that Eve SUCCEEDS *in the described experiment if the following conditions are met:*

(i) $V_{Eve} = R_{C^*}$.
(ii) *The time that Eve needed to output V_{Eve} after she was presented with C^* was at most $c \cdot t_{max}$.*

Said probability of ϵ is taken over the uniformly random choice of $C^* \in \mathbf{C}$, and the random choices or actions that Eve might take in steps 2a, 2b and 2c.

1) The Value of a Semi-Formal Specification: It is clear that Specification 1 is no consistent formal definition. Too many central aspects remain undefined from a strictly formal perspective (and the authors are well aware of this). For example, it is not specified exactly how the adversary is formalized: Is he a classic probabilistic Turing machine (TM)? He should not be a classical TM, since he must be able to conduct physical actions on the SIMPL system while he has access to it. After all, a classical TM cannot execute such physical actions.

But how else could the adversary be formalized? Currently, there is no existing formal model that could capture all possible physical actions he might perform. In lack of such a model, a formal, consistent definition seems impossible.

But does that mean that we have to confine ourselves with the informal description of Section II-A? This would be quite disadvantageous, since the description does not seem specific enough to capture the essence of SIMPL systems. The exact adversarial attack model is unclear, and there is no thorough specification what the "security" of a SIMPL system means. For example, it is not stated in which sense it shall be infeasible for an adversary to determine the responses of the SIMPL system as quickly as the original system.

The route that we propose in the above Specification 1 is, to some extent, a compromise. We intentionally leave some of the aspects of the definition imprecise; one example is the absence of an exact computational model that underlies the adversary's actions. Nevertheless, we believe that the specification helps to illustrate the exact nature of SIMPL systems more exactly, and allows us to specify a number of security parameters that are central to a SIMPL system's security.

Among other things, the specification can hence help to develop a common language and a communication interface between the developers of SIMPL-based protocols, and the hardware designers of the SIMPL systems themselves. A thorough and well-defined communication between these two groups is essential to securely apply SIMPLs in practice.

C. Properties of SIMPL Systems

Let us now discuss several features of SIMPL systems.

1) Immunity against ϵ-fraction Read-out and Simulation: It follows from Specification 1 that it must be practically impossible to measure the values R_{C_i} of a SIMPL system for more than an ϵ-fraction of all parameters $C_i \in \mathbf{C}$ within time t_{Ph}. Otherwise, Eve could create a lookup table for an ϵ-fraction of all possible values R_{C_i} during step 2b. This would enable her to succeed in the security experiment of Specification 1 with probability greater than ϵ. This implies that the set of possible measurement parameters $\mathbf{C}$ must be very large, preferably exponential in some system parameter.

For the same reasons, it must be impossible for Eve to determine more than an ϵ-fraction of all CRPs within time t_C by exhaustive simulation on the basis of Sim and $D(S)$. This again implies that $\mathbf{C}$ must be very large, and/or that the simulation must be time consuming.

2) Immunity against Cloning: Another consequence of Specification 1 is that previous physical access for time t_{Ph} and computations of time t_C must not allow Eve to build a "clone" S' of S, whose responses R'_{C_i} possess the following properties: (i) $R_{C_i} = R'_{C_i}$ for more than an ϵ-fraction of all $C_i \in \mathbf{C}$, and (ii) the generation of the R'_{C_i} works quickly, i.e., within time $c \cdot t_{max}$.

More precisely, the following three types of clones must be practically infeasible:

- *Physical clones*, i.e., exact physical reproductions of S that show the same challenge-response behavior on the same timescales.
- *Digital clones*, i.e., computer algorithms which numerically generate the same responses as S as fast as S.
- *Functional clones*, i.e., physical systems with a possible different structure or larger lengthscales that generate the same responses as fast as S.

Please note that the non-feasibility of functional clones is a strong and subtle requirement. It implies that there are no physical systems whose fabrication can be better controlled (for example because they operate on larger length scales), and which can emulate S in real-time. The related idea of simulating physical systems with (better controllable) other physical systems has again been discussed first by Feynman in [32].

3) No Secret Information in SIMPLs and the Role of the String X: We stated earlier that the security of SIMPL systems should not depend on the secrecy of some sort of binary information contained in the SIMPL. Even if the adversary knows all details about the internal configuration of the SIMPL system, he shall be unable to break its security. As said earlier, this requirement can be met in practice since even an adversary who knows all details about the system may find it hard to physically build or clone the system.

Specification 1 formalizes this requirement by allowing the adversary to know any bitstring X of length q when trying to imitate the input–output behavior of the system. If, for example, one would try to construct a SIMPL by using a digital system with some secret key of length q, then the adversary could succeed in the experiment with probability one by using this key as the additional input X. No such digital, secret key based system can therefore serve as a SIMPL system in our sense.

4) Constant vs. Super-polynomial Time Gap: The time gap between Eve and the real SIMPL system S is required to be at least a constant factor $c > 1$ in Specification 1. This seems surprising, since one might expect the stipulation of an exponential gap here. Still, there are some good reasons for our choice. First, SIMPL systems with a small, constant speed advantage seem easier to realize in practice than systems with larger gaps, leaving alone systems with exponential margins. Secondly, it is unclear whether SIMPLs with an exponential time margin between Eve and the SIMPL exist at all. The only known realistic computational systems which might outperform Turing architectures by a super-polynomial factor are quantum computers [52]. But standard quantum computers possess no immunity against physical cloning. They could be mass-fabricated with the same functionality, and therefore appear unsuited as SIMPL systems. Third, it has been frequently hypothesized within the computational complexity community that there are no realistic hardware systems at all that solve NP-complete problems efficiently in practice. Two recent sources in this context are [50], [51]. This further delimits the hope of SIMPL systems which possess an exponential security margin over Eve.

Fortunately, many applications of SIMPL systems do not require exponential speed gaps. The protocols we suggest in this paper show that a constant, detectable time difference suffices in order to implement such various tasks as identification, message authentication, coin flipping, bit commitment, and zero-knowledge proofs. An exponential time gap between the SIMPL system and any simulation machine is even undesirable for these protocols, since it would lead to too time consuming simulation steps for the honest protocol participants.

5) Feedback Loops: In order to create larger time margins, the absolute, but not the relative (!) time difference between the original SIMPL system and any fraudster can be amplified via feedback loops. Such feedback-loops can be constructed as follows: Presented with a challenge C_1, the SIMPL systems successively determines a sequence of k challenge-responses-pairs $(C_1, R_{C_1}), (C_2, R_{C_2}), \ldots, (C_k, R_{C_k})$, where later challenges C_n are determined by earlier results R_{C_m}, with $k \geq n > m \geq 1$. The tuple (C_1, R_{C_k}) is then regarded as the overall challenge-response pair of the SIMPL system; see [19] for further details. This strategy can amplify the absolute time margin between the SIMPL and the simulator and compensate network and transmission delays.

A concrete example will probably illustrate our point best. Let us assume that we possess a SIMPL system S which produces its responses in t_{max} of 10 nanoseconds (ns), and which possesses a speed advantage of $c = 2$ over all simulations. Any adversaries then cannot produce the response to a randomly chosen challenge within 20 ns. This tiny difference of 10 ns vs. 20 ns would not be detectable in many practical settings, for example in networks with natural delays. Nevertheless, the application of repeated feedback loops can amplify not the relative, but the absolute time margin, to values such as 1 millisecond (ms) vs. 2 ms, or 1 sec vs. 2 sec. These values allow compensation of small transmission delays.

6) SIMPLs with Multi-bit Output: In some applications, it is convenient if a SIMPL system produces not just one bit as response, but a multi-bit output. Some implementations of SIMPLs have this property naturally (for example the optical implementation of section VII-C). Otherwise, feedback loops can allow us to create multi-bit outputs from SIMPL systems with 1-bit outputs: One simply considers a concatenation (or some other function, for example a hash function) of the last n responses $R_{C_{k-n+1}}, \ldots, R_{C_k}$ in the feedback loop. This concatenation (or function) can be interpreted as the overall output of the SIMPL.

Another option to create "large" SIMPL systems with k-bit outputs from "small" SIMPL systems with 1-bit outputs is to employ k such SIMPL systems in parallel, and to directly concatenate their responses to produce a k-bit overall output. This method has been suggested already in the context of PUFs in [13].

7) A Digital Quasi-SIMPL (Which Does not Meet Specification 1): It may be useful for the readers to attempt to design digital, secret key based systems that have some of the properties of SIMPL systems. We call such systems quasi-SIMPLs. One possibility to construct a quasi-SIMPL is as

follows: One takes a private key, public key pair (sk, pk) from a standard digital signature scheme, stores the secret key sk in a hardware system, and makes pk public. Upon receiving a challenge C, the hardware chooses a random number r of length k (with k being a public security parameter), and computes the hardware's response as $R_C = Sig_{sk}(C\|r)$ ($\|$ denoting concatenation). In order to verify that a certain response R_C is correct, one must test by exhaustive search if R_C is a correct signature of the string $C\|r$ for some bitstring r of length k. Choosing k of the correct length will create the desired speed gap.

If the key sk is stored safely in the hardware system, then — seen merely from the outside — it will behave similar as a SIMPL system, i.e., as a quasi-SIMPL. Nevertheless, we would like a true SIMPL system to be free of any secret key information; it would be desirable if Specification 1 ruled out quasi-SIMPLs. And indeed it does: setting the string $X = sk$ allows Eve to succeed in the security experiment of Specification 1 with probability 1. This again illustrates the usefulness of the specification, and stresses the important function of the string X within the specification.

8) Error Correction: Finally a quick note on error correction. In Specification 1 and throughout the rest of the paper, we assumed for the simplicity of our treatment that the responses of a SIMPL system are stable. In practice, error correction must and can be applied to achieve this goal. Reliable information extraction from noisy PUF responses has been treated, for example, in [9], [54], [55], [56], [57] and the references therein. We refer the reader to the large body of existing work on this topic, and ignore error correction aspects in the rest of the paper.

III. Identification and Message Authentication

We now proceed to several cryptographic protocols that can be implemented by SIMPL systems, starting with the identification of entities and the authentication of messages.

A. Identification of Entities

We assume that Alice holds an individual $(t_{max}, c, t_C, t_{Ph}, q, \epsilon)$-SIMPL system S, and has made the corresponding data $D(S)$, Sim, the value $c \cdot t_{max}$, and a description of $\mathbf{C}$ public. Now, she can prove her identity to an arbitrary second party Bob as follows, with k being the security parameter of the protocol:

Protocol 2: Identification of Entities

1) Bob chooses k challenges $C_1, \ldots, C_k$ uniformly at random from $\mathbf{C}$.
2) **For** $i = 1, \ldots, k$ **do**:
   a) Bob sends the value C_i to Alice.
   b) Alice determines the corresponding response R_{C_i} by an experiment on her SIMPL system S, and sends this value to Bob.
   c) Bob receives an answer from Alice, which we denote by V_i. If Alice's answer did not arrive within time $c \cdot t_{max}$, then Bob sets $V_i = \perp$ and continues the for-loop.
3) Bob computes the value $R^{Sim}_{C_i} = \mathsf{Sim}(C_i, D(S))$ for all $i = 1, \ldots, k$, and verifies if $R^{Sim}_{C_i} = V_i \neq \perp$. If this is the case, Bob believes Alice's identity, otherwise not.

In a nutshell, the security of the protocol follows from the fact that an adversary is unable to determine the values R_{C_i} for randomly chosen C_i comparably quickly as Alice. This holds as long as (i) the lifetime of the system S (and the period since $D(S)$ was made public) does not exceed t_C, and (ii) the adversary's accumulated physical access times do not exceed t_{Ph} (see Specification 1). In that case, the adversary's probability to succeed in the protocol without possessing S decrease exponential in k.

Bob can improve his computational efficiency by verifying the correctness of the responses R_{C_i} only for a randomly chosen subset of all responses. If necessary, possible network and transmission delays can be compensated for by amplifying the absolute time gap between Eve and S through feedback loops (see Section II-C5).

If the SIMPL system has multi-bit output (see Section II-C6), then a value of $k = 1$, i.e., a protocol with one round, may suffice. In these cases, the parameter ϵ of the multi-output SIMPL system will in itself be exponentially small in some system parameter (for example in the size of the sensor array in the optical SIMPLs discussed in Section VII-C).

B. Authentication of Messages

Alice can also employ an individual $(t_{max}, c, t_C, t_{Ph}, q, \epsilon)$-SIMPL system S in her possession to authenticate messages to Bob. Again, we suppose that the values $D(S)$, Sim, $c \cdot t_{max}$, and a description of $\mathbf{C}$ are public.

Protocol 3: Authentication of a Message N

1) Alice sends the message N that shall be authenticated to Bob.
2) Bob chooses $k \cdot l$ challenges $C^1_1, \ldots, C^1_k$, $C^2_1, \ldots, C^2_k$, $\ldots$, $C^l_1, \ldots, C^l_k$ uniformly at random from $\mathbf{C}$.
3) **For** $i = 1, \ldots, l$ **do**:
   a) Bob sends the values $C^i_1, \ldots, C^i_k$ to Alice.
   b) Alice determines the corresponding responses $R_{C^i_1}, \ldots, R_{C^i_k}$ by experiments on her SIMPL system S.
   c) Alice derives a MAC-key K_i from $R_{C^i_1}, \ldots, R_{C^i_k}$ by a publicly known procedure, for example by applying a publicly known hash function to these values. She sends $MAC_{K_i}(N)$ to Bob.
   d) Let us denote the answer Bob receives from Alice by V_i. If V_i did not arrive in time $c \cdot t_{max} + t_{MAC}$, where t_{MAC} is the time to derive K_i and compute $MAC_{K_i}(N)$, then Bob sets $V_i = \perp$ and continues the for-loop.
4) For $i = 1, \ldots, k$ and $j = 1, \ldots, l$, Bob computes the values $R^{Sim}_{C^j_i} = \mathsf{Sim}(C^j_i, D(S))$ by simulation via Sim. He derives the keys $K^{Sim}_1 \ldots, K^{Sim}_k$ by application of

the same procedure (e.g. the same publicly known hash function) as Alice in step 3c.

5) For all $i = 1, \ldots, k$, Bob checks if it holds that $MAC_{K_i^{Sim}}(N) = V_i \neq \perp$. If this is the case, he regards the message N as properly authenticated, otherwise not.

The idea behind the protocol is that an adversary cannot determine the responses $R_{C_i^j}$ and the MAC-Keys $K_1, \ldots, K_l$ as quickly as Alice. As earlier, verification of a randomly chosen subset of all MACs can improve Bob's computational efficiency in step 5. Depending on the exact circumstances, a few erroneous V_i may be tolerated in step 5, too.

We assume without loss of generality in Protocol 3 that the MAC can be computed quickly (including the derivation of the MAC keys $K_1, \ldots, K_l$), i.e., within time t_{MAC}, and that t_{MAC} is small compared to $c \cdot t_{max}$. Again, this condition could be realized by amplification through feedback loops if necessary (see Section II-C5). It is known that MACs can be implemented very efficiently [36]. If information-theoretically secure hash functions and MACs are used, the security of the protocol will not depend on any assumptions other than the security of the SIMPL system.

If the SIMPL system has a multi-bit output, then values of $k = 1$, i.e., sending just one challenge in each round, or of $l = 1$, i.e., employing just one round of communication, may suffice. Such a multi-bit output can arise either naturally, for example through the choice of the SIMPL system itself (as noted earlier, the optical SIMPL system mentioned in Section VII-C has this property). Or it can be enforced by feedback loops, or by using several independent SIMPL systems in parallel (see Sections II-C5 and II-C6). In fact, such measures even are strictly necessary to uphold the protocol's security if the constant c has got a very low value.

IV. Two-Player Protocols

SIMPL systems also have a notable potential for two-player protocols. This extends their application potential, but had not been addressed in earlier publications. Three important protocols are covered in this section.

A. Coin Flipping

Coin flipping [33] is a long known two-player protocol which can serve well as a first simple touchstone for the potential of SIMPLs with respect to two-party schemes. Its basic setting is as follows: Two players Alice and Bob want to communicate over a binary channel in order to produce a random binary value B ("a fair coin") as output. The protocol must guarantee that the output cannot be biased or pre-determined by one of the players; see [33] and [46] for more details.

In our setting, we assume that Alice holds a $(t_{max}, c, t_C, t_{Ph}, q, \epsilon)$-SIMPL system with description $D(S)$, and that Bob knows $D(S)$, Sim, and $\mathbf{C}$. Without loss of generality, we assume that the responses of S have a length of one bit (otherwise, one can take the exclusive or of all single bits in the response string, or apply another suited function to the responses). Under these circumstances, a time-restricted coin flipping protocol based on SIMPL systems can be implemented as follows:

Protocol 4: Coin Flipping

1) Alice sends a randomly chosen challenge $C \in \mathbf{C}$ to Bob.
2) Bob immediately after receipt of C answers by sending a random bit r to Alice.
3) Alice verifies if she received r within time less than $c \cdot t_{max}$ after she sent C. If not, she aborts the protocol. Otherwise, she determines R_C by measurement on S, and sets the flipped coin to be $B = R_C \oplus r$.
4) Bob verifies if $C \in \mathbf{C}$, and aborts if this is not the case. He determines R_C by simulation, and sets the flipped coin to be $B = R_C \oplus r$.

The security of the protocol straightforwardly follows from the assumption that S is a $(t_{max}, c, t_C, t_{Ph}, q, \epsilon)$-SIMPL system: If Alice receives the value r within time $c \cdot t_{max}$, then Bob cannot know R_C before he sends away r. He hence cannot choose r as a function of R_C in order to bias the outcome of B. Protocol 4, for the first time, illustrates a potential for two-player protocols in SIMPLs which goes beyond the classical identification and message authentication applications.

B. Bit Commitment

Can more advanced two-party protocols be realized on the basis of SIMPL systems? One good candidate to investigate is bit commitment (BC) [45], [46].

BC is a two-player protocol where one party acts as the sender, and a second party acts as the receiver. The sender holds a bit b at the beginning of the protocol, while the receiver holds the empty input. The protocol has two stages, a commit phase and a reveal phase. At the end of the commit phase, the sender and receiver must have interacted in such a way that the sender has bound or committed himself to the bitvalue b by the communication, but that the receiver does not know this value, and finds it infeasible to derive it from the communication. In the reveal phase, the sender "opens" his commitment and allows the receiver to learn b. After completion of the commit phase, it must be infeasible for the sender to change the commitment he made, and to run the reveal phase in such a way that the receiver learns a different bit $1-b$. Further details and a formal definition can be found in [46]. Bit commitments are important components of zero-knowledge proofs [47], [48], and other, more general two-party cryptographic protocols [49]; see again [46] for further information.

The SIMPL-based BC scheme we suggest here employs interactive hashing (IH) [42] as a sub-protocol. IH is another useful two-player protocol, in which Alice's initial input is an m-bit string C, and Bob has no input. At the end of the protocol, Alice and Bob know two m-bit strings C_0 and C_1, with the properties that (i) $C_j = C$ for some bit $j \in \{0, 1\}$, but Bob does not know the value of j, and that (ii) the other string C_{1-j} is a random bitstring of length m, which neither Alice nor Bob can determine alone. Secure IH can be realized

in an information theoretic fashion, i.e., independently of any computational or other unproven assumptions. For further details, see [42], [43], [44].

In the following Protocol 5, Alice acts as the sender and Bob as the receiver of the bit b. We assume that Bob holds a $(t_{max}, c, t_C, t_{Ph}, q, \epsilon)$-SIMPL system S, and that Alice knows $D(S)$, Sim and $\mathbf{C}$, and holds a bit b she wants to commit. The protocol splits in a commit phase and a reveal phase, and works as follows.

Protocol 5: BIT COMMITMENT

Commit Phase:

1) Alice chooses a random challenge C from $\mathbf{C}$, and determines R_C by simulation.
2) Alice and Bob start an interactive hashing protocol. Alice's input is C, and Bob's input is the empty string. Both get two strings C_0 and C_1 as output.
3) Alice determines the index i for which $C_i = C$, and sends Bob the value $i \oplus b$.

Reveal Phase:

4) Alice sends Bob the values i and R_{C_i} (which is equal to R_C if Alice behaves honestly, and hence known to her from step 1).
5) Bob checks if the time interval between the start of the IH protocol in step 2 and the reception of the values i and R_C in step IV-B is smaller than $c \cdot t_{max}$. If this is the case, he verifies by measurement on S that the value R_{C_i} sent by Alice is correct. If this holds, too, he accepts the BC as valid, and reveals the committed bit by computing $(i \oplus b) \oplus i = b$.

Please note that the commit phase and the reveal phase of this scheme must be executed relatively closely after each other. In particular, Alice must not have time to compute the value $R_{C_{1-i}}$ in the time interval between completion of the interactive hashing protocol in step 2 and the reveal step 4. If she could compute $R_{C_{1-i}}$, she can open the commitment at will by sending either the values i and R_{C_i}, or the values $1-i$ and $R_{C_{1-i}}$ in step 4.

This means that the so-called binding property of the above BC scheme (i.e., the fact that Alice cannot change the value anymore after the commit phase) is conditional upon the prompt execution of the reveal phase. On the other hand, the so-called hiding property of the scheme (i.e., the fact that Bob will not learn b unless the reveal phase is executed) is unconditional: No matter how much time passes, Bob cannot learn the bit b unless Alice gets engaged in the reveal phase.

This implies that if the protocol fails to be executed within said time limits (for example, because the network is down, or other delay occurs), it can be restarted arbitrary many times without endangering the confidentiality of Alice's bit b. The time restriction will therefore not constitute a severe disadvantage in many settings.

C. Zero-Knowledge Proofs

Zero-knowledge proofs (ZK proofs) [47], [48] are a very powerful two-party scheme, in which one party acts as the so-called prover, the other as the so-called verifier. The setting is as follows: The prover is in possession of a solution W to a computationally hard problem Π (for example, a three-coloring of a certain, publicly known, hard graph G), and wants to prove to the verifier that he indeed knows such a solution W to Π — but without revealing W to the verifier. For further details, see [47], [48], [46]. Some application examples of ZK proofs are passwords schemes and authentication systems, as well as the enforcement of honest behavior in cryptographic protocols while maintaining the privacy of the users. Along these lines, they are an essential component in secure multi-party computations [34], [46].

In the following, we give a ZK proof for the three-coloring of a graph that rests on the above SIMPL-based BC protocol. By a well-known reduction result [46] and the NP-completeness of the three-coloring problem, this implies that there are SIMPL-based ZK proofs for all languages in NP. Our proof again employs interactive hashing as a subprotocol; see Section IV-B. In our protocol, we assume that a finite graph $G = (V, E)$ with $V = \{1, \ldots, n\}$ is public, and that Alice knows a three coloring $W : V \to \{00, 01, 11\}$ for this graph. Furthermore, we suppose that Bob holds a $(t_{max}, c, t_C, t_{Ph}, q, \epsilon)$-SIMPL system S, and that Alice knows $c \cdot t_{max}$, $D(S)$, Sim and $\mathbf{C}$. Finally, without loss of generality we assume that the output of S are one-bit values (otherwise, one can take for example the XOR of all output bits to obtain one-bit responses, or apply another suitable function to the output bits).

Protocol 6: ZK PROOF OF A THREE-COLORING W

1) Alice selects $2n$ challenges $C_1, \ldots, C_{2n}$ at random, and determines $R_{C_1}, \ldots, R_{C_{2n}}$ by simulation.
2) Alice selects a random permutation π over $\{00, 01, 11\}$, and forms the string $L = \pi(W(1)) \cdot \pi(W(2)) \cdots \pi(W(n))$.
3) Alice and Bob run $2n$ interactive hashing protocols. In the i-th protocol, Alice's input is C_i, and Alice's and Bob's output is C_i^0, C_i^1. We denote by $k_i \in \{0, 1\}$ the index for which $C_i = C_i^{k_i}$, and define K as $K = k_1 \cdot k_2 \cdots k_{2n}$.
4) Alice sends the string $X = X_1 \cdots X_{2n} = L \oplus K$ to Bob.
5) Bob at random chooses an edge $e = (l, m) \in E$ and sends e to Alice.
6) Alice sends the four values $T = k_{2l-1}, U = k_{2l}, V = k_{2m-1}, W = k_{2m}$ and the corresponding responses $R_{C^T_{2l-1}}, R_{C^U_{2l}}, R_{C^V_{2m-1}}, R_{C^W_{2m}}$ to Bob.
7) Bob verifies if: (i) The two vertices of the edge e are colored differently. He does so by checking whether $(X_{2l-1} \oplus k_{2l-1}) \cdot (X_{2l} \oplus k_{2l}) \neq (X_{2m-1} \oplus k_{2m-1}) \cdot (X_{2m} \oplus k_{2m})$. (ii) The purported responses $R_{C^T_{2l-1}}, R_{C^U_{2l}}, R_{C^V_{2m-1}}, R_{C^W_{2m}}$ are correct. He does so by measurement on S. (iii) The time that passed between step 3 and step 6 is at most $c \cdot t_{max}$. If (i) to (iii) hold, Bob accepts this run of the protocol as successful.

The protocol has an error rate of up to $1 - 1/|E|$. As

usual, polynomially many independent runs can downscale this error rate to any desired value [46]. As noted earlier, it can be observed that if a single run of the protocol fails to be executed within the required time limits (for example, because the network is down), the confidentiality of Alice's three-coloring W is still maintained. This is guaranteed by the fact that the SIMPL-based bit commitment scheme of Protocol 5 is unconditionally hiding.

V. Key Exchange

Secure key exchange is another central cryptographic task in which SIMPL systems and Public PUFs can assist us. We treat this topic at the end of our protocol discussion for two reasons: First of all, we use for the first time material that was originally introduced by others (namely Protocol 7); and second, because one suggested scheme (Protocol 8) builds on the message authentication method of the earlier Section III-B.

A. Key Exchange via PPUFs

As noted in Section I-F, PPUFs [24] are an essentially equivalent concept to SIMPLs. One application suggested in [24] is a key exchange scheme. It requires a special type of SIMPL system, which we call a PPUF, giving honor and credit to [24].

Let S be a $(t_{max}, c, t_C, t_{Ph}, q, \epsilon)$-SIMPL system, and let the function F_S implemented by S fulfill the following additional properties:

(i) F_S is a one-to-one function.
(ii) F_S is a one-way function, i.e., it is hard to invert.
(iii) The time gap c between any simulation and the real-time behavior of S is very large (examples discussed later on require orders of $c > 10^5$ or similar magnitudes).

Under these circumstances, we call S a $(t_{max}, c, t_C, t_{Ph}, q, \epsilon)$-PPUF. Implementations of such systems have been suggested in [24].

On the basis of a PPUF, we can implement a key exchange scheme as described in Protocol 7. Before giving the protocol, we stress once more that the protocol has originally not been devised by us, but is an abstraction from the concrete setting of [24] (i.e., from the concrete PPUF implementation that is used there).

We assume that Alice holds the PPUF S and that Bob knows the corresponding sets and algorithms $D(S)$, Sim and $\mathbf{C}$.

Protocol 7: Key Exchange with PPUFs

1) Bob chooses at random a subset $\mathbf{U}$ of the set of all possible challenges $\mathbf{C}$, with the property that $\mathbf{U}$ can be characterized by a short string I_U.
2) Bob chooses k random challenge $C_1, \ldots, C_k$ from $\mathbf{U}$. He derives a key K from $C_1, \ldots, C_k$ by a publicly known procedure (e.g., a hash function), and determines $R_{C_1}, \ldots, R_{C_k}$ by simulation of S.
3) Bob sends $I_U, R_{C_1}, \ldots, R_{C_k}$ to Alice.
4) Alice uses the PPUF S for a simple exhaustive search in order to find $C_1, \ldots, C_k$: She applies all possible challenges $C' \in \mathbf{U}$ to the PPUF, and compares the response to $R_{C_1}, \ldots, R_{C_k}$. If it matches R_{C_i}, she has found C_i. She derives the same key K from the responses by using the same publicly known procedure as Bob.

Depending on the exact PPUF S that is in use, examples for suitable choices for the sets $\mathbf{U}$ could be the set of all challenges in $\mathbf{C}$ that start with a certain substring; sets of the form $\mathbf{U} = \{x_0, ..., x_0 + n\}$, where x_0 and n are natural numbers; or sets of the form $\mathbf{U} = \{H(x) \,|\, x \in \{x_0, ..., x_0 + n\}\}$, where x_0 and n are natural numbers, and H is a publicly known hash function. The latter choice for $\mathbf{U}$ has been employed in the original protocol of [24]. It possesses several advantages, such as distributing the challenges somewhat randomly within $\mathbf{C}$.

1) Discussion and Analysis: Note that S and F_S must really fulfill the properties (i) to (iii) stated in Section V-A in order to make the protocol work: If F_S was not one-to-one, then the determination of the C_i is ambiguous; Alice's and Bob's keys will not match. Secondly, if F_S was not one-way, then an adversary could eavesdrop the communication, learn $R_{C_1}, \ldots, R_{C_k}$, invert F_S in order to learn $C_1, \ldots, C_k$, and thus derive K. Finally, if feature (iii) is not fulfilled, an adversary Eve could *by numerical simulation* perform the same exhaustive search as Alice in order to identify the values $C_1, \ldots, C_k$ relatively efficiently (see also below). Properties (i) to (iii) therefore are necessary requirements. This is in opposition to earlier protocols, where the employed SIMPL system does not need to fulfill (i) to (iii), making their hardware implementation easier. For example, Protocols 2 to 6 could work with SIMPLs with small time gaps c.

We now analyze the security margin of the protocol in more detail (compare [24]). Let us assume that Bob can simulate the PPUF's response on any challenge in time t_{sim}. As follows from Specification 1, $c \cdot t_{max} \leq t_{sim}$. Furthermore, Specification 1 implies that Alice can execute her measurement on S in time t_{max}, and any adversary Eve requires at least time $c \cdot t_{max}$ in order to simulate the PPUF's response to a randomly chosen challenge.

It therefore holds for Alice's expected workload W_A and Bob's workload W_B in the above protocol that $W_A \approx t_{max} \cdot k/(k+1) \cdot |\mathbf{U}|$, and $W_B \approx t_{sim} \cdot k \geq c \cdot t_{max} \cdot k$. On the other hand, an adversary Eve who numerically simulates all responses $C \in \mathbf{U}$, and who can simulate one response in time $c \cdot t_{max}$, has an expected workload of $W_E \approx c \cdot t_{max} \cdot k/(k+1) \cdot |\mathbf{U}|$. Note that the factors $k/(k+1)$ come in due to standard probability theory as we consider expected workloads.

Thus, the relative advantage of Alice over an adversary who applies the above simple attack strategy of exhaustive search, is $W_E/W_A \approx c$, or

$$W_E \approx W_A \cdot c. \tag{1}$$

In other words, Eve's workload is only separated by the SIMPL system's constant c from the workload of Alice. In order to achieve a long term security of the key, this requires a very large c or substantial values for W_A. Let us consider a few examples: If we stipulate that W_E is required to be on the order of 100 years for security reasons, then $c = 10^5$

makes a workload of $W_A \approx 8.76$ hours necessary for Alice; $c = 10^7$ implies $W_A \approx 5.3$ min; and in order to achieve $W_A \approx 0.3$ sec, a time gap of $c = 10^{10}$ is required. It seems yet uncertain if such large time gaps can be achieved by practical and inexpensive hardware implementations of SIMPL systems; an alternative method that requires only smaller values for c is described in the upcoming Section V-B.

Finally, we note that protocol in practice requires an authenticated channel, which can either be realized by classical means, or by SIMPL/PPUF-based message authentication a la Protocol 3.

B. Authenticated Key Exchange by SIMPLs and Diffie-Hellman

An alternative approach to Protocol 7 is to combine the Diffie-Hellman key exchange protocol with the SIMPL-based message authentication scheme of Protocol 3. This presumes that Alice holds a $(t^A_{max}, c^A, t^A_C, t^A_{Ph}, q^A, \epsilon^A)$-SIMPL system S_A, Bob holds a $(t^B_{max}, c^B, t^B_C, t^B_{Ph}, q^B, \epsilon^B)$-SIMPL system S_B, and that both know the respective values $D(S_A), D(S_B)$, c^A, c^B, t^A_{max}, t^B_{max}, and the algorithm Sim. The protocol is straightforward, but we include it for reasons of completeness.

Protocol 8: AUTHENTICATED KEY EXCHANGE BY SIMPLS AND DH (SCHEMATIC)

1) Alice chooses a random exponent a. She sends the message g^a to Bob, authenticated by use of her SIMPL System S_A and Protocol 3.
2) Alice chooses a random exponent b. He sends the message g^b to Bob, authenticated by use of his SIMPL System S_B and Protocol 3.
3) Both form the exchanged key as $K = g^{ab}$.

One asset of Protocol 8 is that it inherits its long-term security and its authenticated channel from two different sources. It can be carried out efficiently (if SIMPLs with small c^A, c^B and t^A_{max}, t^B_{max} are used), and can hence be employed for the ad-hoc exchange of session keys in communication networks. These keys can be erased whenever needed, being in line with our overall goal of avoiding the long term-presence of secret keys in hardware.

The long-term confidentiality of the protocol, on the other hand, is derived from the well-established Diffie Hellman (DH) assumption. It establishes a large, asymptotically exponential security margin between the computational effort that must be invested by the honest parties to run the protocol and by the adversary to obtain the exchanged key.

Please note in this context that the DH function is a digital function that is optimized in terms of its security properties. It does not need to fulfill any other, possibly involved criteria. Contrary to that, the function implemented by the PPUF/SIMPL in Protocol 7 must be a non-invertible function, similar to the DH function. But in addition, it must depend on unclonable random analog features of the hardware, be stable against environmental conditions and aging, and must be vastly faster than any digital simulator. We feel that this agglomeration of features could potentially become problematic, and that the simulation gap of SIMPLs/PPUFs might be overstretched when it is used to establish the long-term security of a key or the long-term confidentiality of data.

In our opinion, Protocol 8 thus constitutes a viable, at times preferable alternative to Protocol 7,

VI. APPLICATIONS OF SIMPL SYSTEMS

A. Secure Communication Infrastructures

Within the given space restrictions, we will now discuss the application of SIMPL systems to secure communication in networks, illustrating their potential in such a setting. Consider a situation where k parties $P_1, \ldots, P_k$ and a trusted authority TA participate in a communication network. Assume that each party P_i carries its own SIMPL S_i in its hardware, and that a certificate C_i has been issued for each party by the TA. The certificate includes the identity and the rights of Party P_i, and has the form

$$C_i = \big(Id_i, Rights_i, D(S_i), Sig_{TA}(Id_i, Rights_i, D(S_i))\big).$$

Under these provisions, the parties can mutually identify themselves by Protocol 2, they can establish authenticated channels with each other by Protocol 3. They can exchange session keys via the use of the Protocol 8 (or, alternatively, Protocol 7). The whole architecture works without permanent secret keys, or without any other secret information that is stored permanently in the hardware of the parties $P_1, \ldots, P_k$.

It also seems well applicable to cloud computing: All personal data could be stored centrally. Session keys could be exchanged by the Diffie-Hellman protocol over channels authenticated by the SIMPL systems (Protocol 8). These keys can be used to download the personal data in encrypted form from the central storage. The keys can be new in each session, no permanent secret keys in the mobile hardware are be necessary.

The above approaches can further be combined with tamper-sensitive SIMPL systems. These SIMPLs may cover hardware which has a functionality $Func_i$ as long as it is non-manipulated. Each certificate C_i could then also include the functionality of the hardware, i.e., it could be of the form

$$\begin{aligned} C_i \;=\; & \big(Id_i, Rights_i, Func_i, D(S_i), \\ & Sig_{TA}(Id_i, Rights_i, Func_i, D(S_i))\big). \end{aligned}$$

By running the identification protocol (Prot. 2), party P_i can prove that the SIMPL system S_i is non-tampered, and that the hardware hence has the claimed functionality $Func_i$. Please note that the optical SIMPL systems we propose in this paper is naturally tamper sensitive; the tamper sensitivity of such optical scattering structures has already been shown in detail in [8].

Finally, by using Protocols 4, 5 and 6, all parties can execute several typical two-party computations with each other, leading to various further cryptographic applications.

B. Two other Applications: Unforgeable Labels and DRM

Let us in all brevity sketch to two other applications of SIMPL systems, which have been described in more detail in [16].

The first of these applications is the generation of unforgeable labels for products or security tokens. SIMPL systems can create labels which do not contain any secret information, which can be verified offline, and which only require remote, digital communication between the label and a testing device.

SIMPL systems can be applied in this context. A SIMPL-label consists of the following components: (i) The SIMPL System S; (ii) The description $D(S)$ and some product related info I; and (iii) the digital signature $Sig_{SK}(D(S), I)$, created by the secret signing key SK of the label issuer. Components (ii) and (iii) are digital information that can be stored on the labeled item of value, for example via a printed barcode or electronic means.

In the verification of a label, a testing apparatus obtains $D(S)$ from the label, verifies the digital signature via use of a publicly known verification key PK, and executes Protocol 2 in order to check the presence of the SIMPL system S. A description of $\mathbf{C}$, t_{max} and Sim need to be hardwired into the apparatus together with PK. If more than one label issuer is involved, the apparatus can store more than one public verification key, or standard signed key certificates can be employed.

Labels based on SIMPL system have interesting advantages: They can be read out digitally and remotely. Secondly, they can be verified be offline, i.e. without an online connection to a central institution/database. The labels do not contain any secret information at all, also not in the form of a PUF. Finally, also the testing apparatus that evaluates the validity of a label does not need to contain any form of secret information. The only secret key involved in the scheme remains centrally with the issuer of the label, where it can be well protected. In combination, these features distinguish SIMPL-based labels from other known approaches.

Note that the issuer of a SIMPL-based labels can create the required signature of component (iii) remotely, i.e., he does not need to be present at the production site where the label is generated and attached to the item of value. His secret signing key can be kept to him alone. This is particularly useful in situations where illegitimate overproduction at remote manufacturing sites must be encountered.

Another application area of SIMPLs lies in the context of the digital rights management problem (DRM). Similar to the above labels, SIMPLs can also create unclonable representations of digital content, including software [16]. These unclonable representations do not contain any secret information, and can be verified by a testing device that does not need to contain any secret keys either. The verification works offline and by mere digital communication between the testing device and the device carrying the unclonable representation. Again, in combination these features are not met by any comparable technique known to the author. In [38], [39], [40], for example, the random features of the data carrier must be determined in the near-field by analog measurements. The features must be communicated correctly by the analog measurement apparatus (e.g., the optical drive) to a central module (e.g., a TPM) that decides about the validity of the content, meaning that the measurement apparatus must be trusted.

VII. Implementation of SIMPL Systems

We now turn to the practical implementation of SIMPL systems. Our aim is to give an overview of the particular challenges in the realization of SIMPLs and the existing implementation candidates, and to refer the reader to the existing literature for the details of the described approaches.

A. Challenges

There are some clear challenges in the realization of SIMPL systems. Three non-trivial requirements that must be balanced are complexity, stability, and simulatability: On the one hand, the output of a SIMPL system must be sufficiently complex to require a long computation/simulation time. On the other hand, it must be simple enough to allow simulation at all, and to enable the determination of $D(S)$ by measurement or numeric analysis techniques. A final requirement is that the simulation can be carried out *relatively* efficiently by everyone (this is necessary to complete the verification steps in the identification and message authentication protocols quickly); while, at the same time, even a very well equipped attacker, who can potentially attempt to parallelize the simulation on many powerful machines, cannot simulate as fast as the real-time behavior of the SIMPL system. In the sequel, we list several implementations that show potential to meet these demanding requirements.

B. Electrical SIMPL Systems

Since the first publication of [16], a sequence of papers of our group has dealt with the implementation of SIMPL systems by electrical, integrated circuits [17], [18], [19], [21]. We tried to exploit two known speed bottlenecks of modern CPUs: Their problems in dealing simultaneously with very large amounts of data, and the complexity of simulating inherently analog and parallel phenomena. Let us briefly summarize these approaches from said papers.

1) "Skew" SRAM Memories: A first suggestion made in [17], [18], [19], [21] is to employ large arrays of SRAM cells with a special architecture named "skew design". In this design, the write behavior of the cells is dependent on the applied operational voltage. If the operational voltage is below a certain threshold, all write operations malfunction. The simulation of many successive read- and write events of the skew SRAM memory under quickly varied operational voltages on a standard architecture then necessarily creates some computational overhead, since in the standard architecture the bit values that are effectively written into the cells must be pre-computed as a function of the operational voltages and the a priori unknown content of the target cell. The hypothesis put forward in [17], [18], [19], [21] is that this creates a small, constant simulation overhead, in particular that it creates the necessity for additional read-operations. Two essential ingredients in this concept are: No parallelization is possible, since the successive read- and write events in the feedback loop are made dependent on the previous read results. And since no parallelization is possible, the limiting factor for an adversary is his clock frequency, which is quite strongly limited by current technology.

As argued in the listed references, the idea shows promise to succeed against any adversaries with a limited financial budget, and in particular to defeat any FPGA-based attacks. Future work will need to characterize how large the exact simulation margin is, and whether it is indeed sufficient to defeat an adversary with strong financial resources who is capable of fabricating ASICs. Due to its relatively easy realizability and good security level, the concept has a good potential for the consumer market.

2) Two-dimensional Analog Computing Arrays: A second suggestion of [17], [18], [19], [21] consists of using analog, two-dimensional computing arrays. The authors suggest the use of so-called cellular non-linear networks (CNNs) which are designed to imitate non-linear optical systems. Due to their analog and inherently parallel nature (many cells exchange information at the same time), CNNs are time consuming to simulate on a digital, sequential architecture. This claim is supported by the standard literature on CNNs, which describes that these analog architectures can outperform classical digital computers by factors of up to 1,000 in certain, specialized tasks like image recognition [22], [23].

The use of CNNs has its assets on the security side: Since it is based on manufacturing mismatches in CNN fabrication that currently seem unavoidable, it could eventually defeat even attackers with very strong financial resources, and has the potential to create SIMPLs that cannot even be clobed by their own manufacturer (i.e., SIMPLs which are manufacturer resistant in the sense of [29]). On the downside, since CNNs are complex analog circuits, they might be less suited for low-cost applications.

3) Other Electrical Approaches: Independently, the work of other groups has lead to different electrical structures that could be used as SIMPLs. The implementation of PPUFs presented in [24] could potentially be downscaled to become a SIMPL system, even though it would have to be carefully investigated how resilient such small-scale instances are against parallelization attacks. Another very interesting, FPGA-based candidate for SIMPLs is implicit in the work of [26].

C. Integrated Optical SIMPLs

A second route that was followed in the implementation of SIMPL systems is the employment of optical structures [16], [20]. The rationale behind this strategy is as follows: First, optical systems can potentially achieve faster component interaction than electronic systems; this promises to create the desired speed advantage over any electronic simulator. In particular, the phenomenon of optical interference has no electronic analog at room temperature [59], and can create a computational overheads. Second, the material degradation of optical systems is low, and their temperature stability is known to be high [59], [60]. Even very complex and randomly structured optical systems, whose internal complexity creates the desired speed gaps, can produce outputs that are relatively stable against aging and environmental conditions.

A concrete optical SIMPL system was suggested in [20]. It comprises of an immobile laser diode array with k phase-locked diodes $D_1, \ldots, D_k$ [61], which is attached to a disordered, random optical scattering medium. The diodes can be switched on and off independently, leading to 2^k possible challenges or inputs C_i to the medium. These challenges can be written as $C_i = (b_1, \ldots, b_k)$, where each $b_i \in \{0, 1\}$ indicates whether diode D_i is switched on or off. Note that the diode array must indeed be phase locked in order to allow interference of the different diode signals. At the oppposite side of the medium, an array of l light sensors $S_1, \ldots, S_l$, e.g. photodiodes, measures the resulting wave front when leaving the scattering medium: It detects the local light intensities at each of the sensors. A response R_{C_i} thus consist of the intensities $I_1, \ldots, I_l$ in the l sensors. Instead of phase-locked diode arrays, also a single laser source with a subsequently placed, inexpensive light modulator (as contained in any commercially available beamer) can be employed.

Under the provision that a *linear* scattering medium is used in such integrated optical SIMPLs, the input/output behavior of this SIPML can be machine learned and predicted. This was shown by a proof of concept implementation in [20]. As argued in the same publication, there is also a time margin between any numeric simulator and real implementations of the system that are optimized with respect to speed: While the real system can create its output pattern in nanoseconds, the simulation requires around $k \cdot l$ additions of precomputed values. For moderate sizes of the system of $k = l = 10^4$, this requires 10^8 precomputed values and 10^8 additions. This can create exactly the notable, constant speed gap between the real system and the simulator that is required in SIMPL systems.

D. Other Implementation Strategies

There are two further promising implementation strategies that could assist us in creating secure future generations of SIMPLs.

1) Employing PUFs with Reduced Complexity: One generic further strategy for the realization of SIMPL systems, which has been suggested already in [16], is the following: Employ a PUF or a PUF-like structure; and reduce its inner complexity until it can be characterized by measurements and simulated, or until it can successfully be machine learned. If the level of complexity is still sufficient, then this simulation will be more time consuming than the real-time behavior of the system. In fact, some suggestions of the previous subsections used this strategy already, since both CNNs and integrated optical structures have already been suggested as PUFs in earlier work [53], [12].

2) Simulation vs. Verification: Another idea is to exploit the well-known asymmetry between actively computing a solution for a certain problem and verifying the correctness of a proposed solution (as also implicit in the infamous P vs. NP question) [16]. Exploiting this asymmetry could lead to protocols of the following kind: A SIMPL system provides the verifier in an identification/authentication protocols with some extra information that allows the verifier to *verify* its answers fast. To illustrate our point, imagine an analog, two-dimensional, cellular computing array whose behavior is governed by partial differential equations (PDEs), such as the CNN described in section VII-B. Then, verifying the

correctness of a given final state of such a PDE-driven system (i.e. verifying that this state is indeed a solution of the PDEs driving the system) could be much more time efficient than computing this solution from scratch. Furthermore, the verifier could not only be given external outputs of such a two-dimensional array (e.g. values in boundary cells), but also internal sub-measurements (e.g. values in inner cells) that help him to verify the output quickly.

The simulation vs. verification strategy can help to relieve the tension between the requirement for fast simulation on the side of the verifier (who may not be well equipped on the hardware side) and the necessary time margin to any attackers (who may be very well equipped on the hardware side), which we already mentioned in Section VII-A.

VIII. Summary, Discussion, and Future Work

A. Summary

This paper introduced and discussed a security concept termed *SIMPL system*. We started out by explaining the basic idea behind this new concept, and developed a semi-formal specification of the exact security properties of SIMPL systems in Section II. Some basic properties that follow from this specification were discussed in the same section, for example the impossibility for cloning a SIMPL system, or for reading out its entire CRP-space. Next, we presented several protocols that can be realized by SIMPL systems in Sections III to V. They include identification, message authentication and key exchange schemes, as well as two-party protocols like coin-flipping, bit commitment, and zero-knowledge proofs of NP-complete languages. We argued that the time restrictions required for these protocols (i.e., the fact that some of them must be executed withint a certain time bound in order to guarantee their security) do not too strongly diminish their practical usability in many relevant settings. Our work reveals the substantial *cryptographic* potential of SIMPL systems, including their application to classical two-party problems, which was previously undiscovered.

Concrete application scenarios of SIMPLs were discussed in Section VI. We described communication infrastructures that work without permanent secret key information in the hardware, and where the hardware can remotely prove its functionality to other parties. Other applications we investigated were unforgeable product labels and digital rights management. In all of these scenarios, SIMPL systems allow us to design cryptographic hardware that does not contain any secret key information, that is, any information whose disclosure breaks the security of the system. This can lead to future generations of hardware that does not require costly protection mechanisms on the physical and software level – there simply is no secret key to protect in SIMPL based hardware. This could make future security hardware more lightweight, mobile and secure at the same time.

Finally, the implementation of SIMPL systems was addressed in Section VII. Due to the large body of existing work, we focused on surveying current implementation candidates, and provided the reader with references to the literature. We covered electrical implementations based on special SRAM memories, two-dimensional analog arrays known as cellular non-linear networks (CNNs), and addressed suggestions by other groups based on circuit glitches and FPGAs. We also pointed to a recent, promising, and integrated optical candidate.

B. Discussion and Analysis

Let us conclude this work by a detailed comparative analysis of SIMPL systems. As said earlier, there are some obvious similarities between classical private/public key cryptoschemes and SIMPL systems: The numeric description $D(S)$ is some analog to a public key, while the physical system S itself constitutes some equivalent to a private key. This provides SIMPLs with a public-key like functionality. It allows new protocols and leads to several practicality advantages, as discussed in previous sections.

Still, there is one important difference to classical, mathematical public-key systems: Our "private key" is no secret number, but a randomly structured, hard-to-clone *physical system*, the SIMPL system S. It has the interesting feature of not containing any form of secret information: Neither in an explicit digital form like a digital key in classical hardware. Nor in a hidden, analog form such as internal PUF parameters (for example the mentioned delay values in Arbiter PUFs, or the parameters determining SRAM behavior in SRAM PUFs). All internal characteristics of a SIMPL, including its precise internal configuration, can be publicly known without compromising the security of the derived cryptographic protocols.

The security of SIMPL systems is not free of assumptions, though. Instead of presupposing the secrecy of some sort of information, it rests on the following two hypotheses: (i) on the computational assumption that no other, well-controllable, configurable, or even programmable hardware can generate the complex responses of a SIMPL with the same speed, and (ii) on the physical assumption that it is practically infeasible for Eve to exactly clone or rebuild the SIMPL system, even though she knows its internal structure and properties. [2]

It is long accepted that computational assumptions play a standard role in mathematical cryptography, and they are also a part of the security assumptions for SIMPL systems; but SIMPLs show that one can trade the need for secret information in the hardware against assumptions on the physical unclonability of the system. This can surprisingly obviate the familiar requirement that cryptographic hardware must contain secret key information of some sort. By the protocols presented in this paper, the communicants can nevertheless execute a very large number of cryptographic protocols and tasks, without employing long-term present secret key information.

C. Future Work and Prospects

Future work on SIMPLs will likely concentrate on developing new protocols for SIMPL systems, and on devising

[2]The reader can verify the plausibility of the latter unclonability property by considering the optical implementation of section VII-C: Even if the positions of all scattering centers and the other irregularities in the scattering medium were known in full detail, it would still be infeasible to rebuild the scattering medium with perfect precision.

formal security proofs for these protocols. For example, it seems interesting if time-restricted, but still useful variants of secure multi-party computation could be implemented by SIMPLs, and how the security of such constructions could be proven. But perhaps the greater challenge lies on the hardware side: Even though there are several promising candidates (see Section VII), the issue of finding a highly secure, practical, and cheap implementation appears not to be fully settled yet. If such an implementation is found, or if the existing implementation candidates are shown to possess all necessary properties, this could potentially change the way we exercise cryptography and security today.

ACKNOWLEDGEMENTS

The author would like to thank Jürg Wullschleger for suggesting the presented coin flipping protocol, and Ulf Schlichtmann, Stefan Wolf, Jürg Wullschleger, Srinivas Devadas, Miodrag Potkonjak and Farinaz Koushanfar for enjoyable and helpful discussions on the general topic of SIMPLs/PPUFs.

REFERENCES

[1] http://www.cbsnews.com/stories/2010/02/15/business/main6209772.shtml
[2] http://www.bbc.co.uk/news/10569081
[3] http://www.eurosmart.com/images/doc/Eurosmart-in-the-press/2006/cardtechnologytoday_dec2006.pdf
[4] http://www.gsaietsemiconductorforum.com/2010/delegate/documents/GASSELGSALondon20100518presented.pdf. Slide 23.
[5] T. Eisenbarth, T. Kasper, A. Moradi, C. Paar, M. Salmasizadeh, M. T. Manzuri Shalmani: *On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme.* CRYPTO 2008.
[6] T. Kasper, M. Silbermann, C. Paar: *All You Can Eat or Breaking a Real-World Contactless Payment System*. Financial Cryptography and Data Security, January 25-28, 2010.
[7] R. J. Anderson: *Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd Edition.* ISBN: 978-0-470-06852-6, Wiley, 2008.
[8] R. Pappu, B. Recht, J. Taylor, N. Gershenfeld, *Physical One-Way Functions*, Science, vol. 297, pp. 2026-2030, 20 September 2002.
[9] R. Pappu, *Physical One-Way Functions*, PhD Thesis, MIT.
[10] B. Gassend, D. E. Clarke, M. van Dijk, S. Devadas: *Silicon physical random functions.* ACM Conference on Computer and Communications Security 2002, 148-160.
[11] B. Gassend, D. Lim, D. Clarke, M. v. Dijk, S. Devadas: *Identification and authentication of integrated circuits.* Concurrency and Computation: Practice & Experience, pp. 1077 - 1098, Volume 16, Issue 11, September 2004.
[12] P. Tuyls, B. Skoric: *Strong Authentication with Physical Unclonable Functions.* In: Security, Privacy and Trust in Modern Data Management, M. Petkovic, W. Jonker (Eds.), Springer, 2007.
[13] G. Edward Suh, Srinivas Devadas: *Physical Unclonable Functions for Device Authentication and Secret Key Generation.* DAC 2007: 9-14
[14] B. Gassend, M. v. Dijk, D. E. Clarke, E. Torlak, P. Tuyls, S. Devadas: *Controlled physical random functions and applications.* ACM Trans. Inf. Syst. Secur. 10(4), 2008.
[15] U. Rührmair: *Oblivious Transfer based on Physical Unclonable Functions (Extended Abstract).* TRUST 2010, Lecture Notes in Computer Science, Volume 6101, pp. 430-440, Springer, 2010.
[16] U. Rührmair: *SIMPL Systems: On a Public Key Variant of Physical Unclonable Functions.* Cryptology ePrint Archive, Report 2009/255, 2009.
[17] U. Rührmair, Q. Chen, P. Lugli, U. Schlichtmann, M. Stutzmann, G. Csaba: *Towards Electrical, Integrated Implementations of SIMPL Systems.* Cryptology ePrint Archive, Report 2009/278, 2009.
[18] Q. Chen, G. Csaba, X. Ju, S.B. Natarajan, P. Lugli, M. Stutzmann, U. Schlichtmann, U. Rührmair: *Analog Circuits for Physical Cryptography.* 12th International Symposium on Integrated Circuits (ISIC2009), Singapore, December 14 - 16, 2009.
[19] U. Rührmair, Q. Chen, M. Stutzmann, P. Lugli, U. Schlichtmann, G. Csaba: *Towards Electrical, Integrated Implementations of SIMPL Systems.* Workshop in Information Security Theory and Practice (WISTP 2010), Passau (Germany), 2010. LNCS, Volume 6033, pp. 277 - 292, Springer Verlag, Berlin, 2010.
[20] U. Rührmair: *SIMPL systems, or: Can we designs cryptographic hardware without any secret key information?* SOFSEM 2011, Lecture Notes in Computer Science, Vol. 6543, Springer, 2011.
[21] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, M. Stutzmann, U. Rührmair: *Circuit-based approaches to SIMPL systems*. Journal of Circuits, Systems, and Computers, JCSC, vol. 20, 2011, pp. 107-123, doi:10.1142/S0218126611007098.
[22] L. O. Chua, T. Roska, T. Kozek, A. Zarandy: *CNN Universal Chips crank up the computing power*. Circuits and Devices Magazine, IEEE, Vol. 12, no. 4, pp. 1828, July 1996.
[23] T. Roska: *Cellular Wave Computers for Nano-Tera-Scale Technology beyond spatial-temporal logic in million processor devices.* Electronics Letters, April 12, 2007, Vol. 43, No. 8.
[24] N. Beckmann, M. Potkonjak: *Hardware-Based Public-Key Cryptography with Public Physically Unclonable Functions.* Information Hiding 2009: 206-220.
[25] Farinaz Koushanfar, Miodrag Potkonjak: *CAD-based Security, Cryptography, and Digital Rights Management*. DAC 2007: 268-269
[26] M. Majzoobi, A. Elnably, F. Koushanfar, *FPGA Time-Bounded Unclonable Authentication*. Information Hiding Conference, pp. 1-16, LNCS 6387, 2010.
[27] U. Rührmair, J. Sölter, F. Sehnke: *On the Foundations of Physical Unclonable Functions*. IACR Cryptology E-print Archive, Report No. 227/2009, 2009.
[28] U. Rührmair, H. Busch, S. Katzenbeisser: *Strong PUFs: Models, Constructions and Security Proofs.* A.-R. Sadeghi, D. Naccache (Editors): Towards Hardware Intrinsic Security: Foundation and Practice. Springer, 2010.
[29] Blaise Gassend, *Physical Random Functions*, MSc Thesis, MIT, 2003.
[30] J. Guajardo, S. S. Kumar, G. J. Schrijen, P. Tuyls: *FPGA Intrinsic PUFs and Their Use for IP Protection.* CHES 2007: 63-80.
[31] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, J. Schmidhuber: *Modeling Attacks on Physical Unclonable Functions*. 17th ACM Conference on Computer and Communications Security, 2010. Previous versions available from Cryptology ePrint Archive, Report 251/2010.
[32] R. P. Feynman: *Simulating physics with computers*. International journal of theoretical physics, Springer, 1982.
[33] M. Blum: *Coin flipping by telephone*. In Proc. IEEE Spring COMPCOM, pages 133–137. IEEE, 1982.
[34] O. Goldreich, S. Micali, A. Wigderson: *How to play any mental game*. Proceedings of the nineteenth annual ACM symposium on th theory of computing, 1987.
[35] O. Goldreich, S. Micali, A. Wigderson: *Proofs that yield nothing but their validity and a methodology of cryptographic protocol design.* 27th Annual Symposium on the Foundations of Computer Science (FOCS), 1986.
[36] S. Halevi, H. Krawczyk: *MMH: Software Message Authentication in the Gbit/Second Rates.* FSE 1997: 172-189
[37] Gerald DeJean, Darko Kirovski: *RF-DNA: Radio-Frequency Certificates of Authenticity.* CHES 2007: 346-363.
[38] Youry Kariakin: *Authentication of Articles*. Patent writing, WO/1997/024699, available from http://www.wipo.int/pctdb/en/wo.jsp?wo=1997024699, 1995.
[39] D. Vijaywargi, D. Lewis, D. Kirovski: *Optical DNA.* Financial Cryptography and Data Security, pp. 222-229, 2009.
[40] G. Hammouri, A. Dana, B. Sunar: *CDs Have Fingerprints Too.* CHES 2009: 348-362.
[41] W. Diffie and M.E. Hellman: *New Directions in Cryptography*. IEEE Transactions on Information Theory, vol. IT-22, Nov. 1976, pp. 644654.
[42] M. Naor, R. Ostrovsky, R. Venkatesan, M. Yung: *Perfect zero-knowledge arguments for NP using any one-way function.* Journal of Cryptology 11(2), 87108 (1998)
[43] G. Savvides: *Interactive Hashing and reductions between Oblivious Transfer variants*. PhD thesis, McGill University, Montreal, 2007.
[44] I. Haitner, O. Reingold: *A new interactive hashing theorem.* IEEE Conference on Computational Complexity, 2007.
[45] M. Blum: *Coin flipping by telephone*. Allen Gersho (editor), Advances in Cryptography, pages 1115, Santa Barbara, California, USA, 1982. University of California, Santa Barbara.
[46] O. Goldreich: *The Foundations of Cryptography – Volume 1*. Cambridge University Press, 2001.

[47] O. Goldreich, S. Micali, A. Wigderson: *Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems.* Journal of the Association for Computing Machinery, 38(3):691729, July 1991.
[48] G. Brassard, D. Chaum, C. Crepeau: *Minimum disclosure proofs of knowledge*. JCSS, 37:156189, 1988.
[49] J. Kilian: *Founding cryptography on oblivious transfer*. In Proc. 20th ACM Symposium on Theory of Computing, pages 20 31, Chicago, 1988. ACM.
[50] Andrew Chi-Chih Yao: *Classical physics and the Church-Turing Thesis*. Journal of the ACM 50(1), 100-105, 2003.
[51] Scott Aaronson: *NP-complete Problems and Physical Reality.* Electronic Colloquium on Computational Complexity (ECCC), 026, 2005.
[52] Peter W. Shor: *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. SIAM J. Comput. 26(5): 1484-1509 (1997)
[53] 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 CNNA, 2010.
[54] D. Lim: *Extracting Secret Keys from Integrated Circuits.* M.Sc. Thesis, MIT, 2004.
[55] G. E. Suh, C. W. O'Donnell, I. Sachdev, and S. Devadas: *Design and Implementation of the AEGIS Single-Chip Secure Processor Using Physical Random Functions*. Proc. 32nd ISCA, New York, 2005.
[56] Meng-Day (Mandel) Yu, Srinivas Devadas: *Secure and Robust Error Correction for Physical Unclonable Functions*. IEEE Design & Test of Computers 27(1): 48-65 (2010)
[57] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Berk Sunar, Pim Tuyls: *Memory Leakage-Resilient Encryption Based on Physically Unclonable Functions*. ASIACRYPT 2009: 685-702
[58] U. Rührmair, A. Weiershäuser, S. Urban, C. Hilgers, J. Finley: *Secure Integrated Optical Physical Unclonable Functions.* In preparation, 2010.
[59] Stephen G. Lipson: *Optical Physics.* 3rd ed., Cambridge University Press, 1995. ISBN 0-5214-3631-1.
[60] Wolfgang Demtröder: *Experimentalphysik 2: Elektrizität und Optik.* Springer 2004. ISBN-10: 3540202102.
[61] D. Zhou, L.J. Mawst: *Two-dimensional phase-locked antiguided vertical-cavity surface-emitting laser arrays*. Applied Physics Letters, 2000.