Protocol Attacks on Advanced PUF Protocols and Countermeasures

Marten van Dijk
University of Connecticut
Storrs, CT 06269, USA
E-mail: vandijk@engr.uconn.edu

Ulrich Rührmair
Technische Universität München
80333 München, Germany
E-mail: ruehrmair@ilo.de

***Abstract*—In recent years, PUF-based schemes have not only been suggested for the basic security tasks of tamper sensitive key storage or system identification, but also for more complex cryptographic protocols like oblivious transfer (OT), bit commitment (BC), or key exchange (KE). These more complex protocols are secure against adversaries in the *stand-alone, good PUF model*. In this survey, a shortened version of [17], we explain the stronger *bad PUF model* and *PUF re-use model*. We argue why these stronger attack models are realistic, and that existing protocols, if used in practice, will need to face these.**

One consequence is that the design of advanced cryptographic PUF protocols needs to be strongly reconsidered. It suggests that Strong PUFs require additional hardware properties in order to be broadly usable in such protocols: Firstly, they should ideally be *erasable*, meaning that single PUF-responses can be erased without affecting other responses. If the area efficient implementation of this feature turns out to be difficult, new forms of Controlled PUFs [3] (such as Logically Erasable and Logically Reconfigurable PUFs [6]) may suffice in certain applications. Secondly, PUFs should be *certifiable*, meaning that one can verify that the PUF has been produced faithfully and has not been manipulated in any way afterwards. The combined implementation of these features represents a pressing and challenging problem for the PUF hardware community.

***Index Terms*—(Strong) Physical Unclonable Functions; (Strong) PUFs; Attack Models; Oblivious Transfer; Bit Commitment; Key Exchange; Erasable PUFs; Certifiable PUFs**

I. Introduction

Today's electronic devices are mobile, cross-linked and pervasive, which makes them a well-accessible target for adversaries. The well-known protective cryptographic techniques all rest on the concept of a secret binary key: They presuppose that devices store a piece of digital information that is, and remains, unknown to an adversary. It turns out that this requirement is difficult to realize in practice. Physical attacks such as invasive, semi-invasive or side-channel attacks carried out by adversaries with one-time access to the devices, as well as software attacks like application programming interface (API) attacks, viruses or Trojan horses, can lead to key exposure and security breaks. As Ron Rivest emphasized in his keynote talk at CRYPTO 2011 [13], merely calling a bit string a "secret key" does not make it secret, but rather identifies it as an interesting target for the adversary.

Indeed, one main motivation for the development of *Physical Unclonable Functions (PUFs)* is their promise to better protect secret keys. A PUF is an (at least partly) disordered physical system P that can be challenged with so-called external stimuli or challenges c, upon which it reacts with corresponding responses r. Contrary to standard digital systems, these responses depend on the micro- or nanoscale structural disorder of the PUF. It is assumed that this disorder cannot be cloned or reproduced exactly, not even by the PUF's original manufacturer, and that it is unique to each PUF. Any PUF P thus implements a unique and individual function f_P that maps challenges c to responses $r = f_P(c)$. Tuples (c, r) are called the *challenge-response pairs (CRPs)* of the PUF.

Due to its complex internal structure, a PUF can avoid some of the shortcomings of classical digital keys. It is usually harder to read out, predict, or derive PUF-responses than to obtain digital keys that are stored in non-volatile memory. The PUF-responses are only generated when needed, which means that no secret keys are present permanently in the system in an easily accessible digital form. Finally, certain types of PUFs are naturally tamper sensitive: Their exact behavior depends on minuscule manufacturing irregularities, often in different layers of the IC. Removing or penetrating these layers will automatically change the PUF's read-out values. These facts have been exploited in the past for different PUF-based security protocols. Prominent examples include identification [12], [4], key exchange [12], and various forms of (tamper sensitive) key storage and applications thereof, such as intellectual property protection or read-proof memory [5], [8], [22].

In recent years, also the use of PUFs in more advanced cryptographic protocols together with formal security proofs has been investigated. In these protocols, PUFs with a very large challenge set and a freely accessible challenge-response interface are employed. This type of PUF sometimes has been referred to as *Physical Random Function* [4] or *Strong PUF* [5], [20], [19], [15] in the literature. The (Strong) PUF is used similar to a *physical random oracle* in these protocols, which is passed on between the parties, and which can be read-out exactly by the very party who currently holds physical possession of it. Its input-output behavior is assumed to be so complex that its response to a randomly chosen challenge cannot be predicted numerically and without direct physical measurement, not even by a person who had physical access to the Strong PUF at earlier points in time.

978-3-9815370-2-4/DATE14/©2014 EDAA

In 2010, Rührmair [14] showed that oblivious transfer can be realized between two parties by physically transferring a Strong PUF in this setting. He observed that via the classical reductions of Kilian [7], this implies PUF-based bit commitment and PUF-based secure multi-party computations. In the same year, the first formal security proof for a Strong PUF protocol was provided by Rührmair, Busch and Katzenbeisser [15]. They present definitions and a reductionist security proof for Strong PUF based identification. In 2011, Rührmair, Jaeger and Algasinger [18] discuss an attack on a PUF-based session key exchange scheme of Tuyls and Skoric [23], in which the scheme is broken under the provision that it is executed several times and that the adversary gains access to the PUF more than once. At CRYPTO 2011 Brzuska, Fischlin, Schröder and Katzenbeisser [1] adapted Canetti's universal composition (UC) framework [2] to include PUFs, giving PUF-protocols for oblivious transfer (OT), bit commitment (BC), and key exchange (KE). At CHES 2012, Rührmair and van Dijk [16] presented a quadratic attack on Brzuska et al.'s OT- and BC-protocols, showing that their security is not maintained if optical PUFs or electrical PUFs with challenge length of 64 bits are used in their implementation. Recent work continues this general line of work: At Eurocrypt 2013 Ostrovsky, Scafuro, Visconti and Wadia [10] investigated the use of so-called malicious PUFs, and furthermore extend Brzuska et al.'s communication model in the UC framework. At IEEE S&P 2013 Rührmair and van Dijk [17] proposed a model equivalent to malicious PUFs under the name *bad PUF model*, and a new attack model (motivated by the attack of [18] that breaks [23]) termed *PUF re-use model*.

This survey is a shortened summary of [17] and explains why the bad PUF model and PUF re-use model are realistic threats and what can possibly be done against these threats. For a detailed discussion on how to exactly break existing protocols in the bad PUF model, PUF re-use model, or a combination of the two, we refer the reader to [17]. The elementary PUF use as internal key storage element and in simple identification protocols [11], [12] seems less affected.

a) Organization of this paper: In Section II we discuss and introduce various attack models for Strong PUF protocols. Section III discusses the need for two counter measures: Erasable PUFs and Certifiable PUFs.

II. Attack Models for Strong PUF Protocols

Section II-A explains the *stand-alone, good PUF model*, where we assume that there is only one single, isolated protocol execution, and that all parties faithfully generate and never manipulate PUF hardware. In Sections II-B and II-C we explain stronger adversarial models:

1) The *PUF re-use model* [18], [17] extends the stand-alone, good PUF model in that adversaries are allowed multiple access to PUFs. Its mildest form is the so-called one-time posterior access model (PAM), which allows one-time access to the PUF after a given protocol, and delimits the adversary to mere CRP-measurement on the PUF.
2) The *Bad PUF model* [17], [10] allows fraudulent parties and adversaries to manipulate PUF hardware and to use so-called bad PUFs: These are PUFs which look like a normal PUF from the outside, having a standard CRP-interface etc., but which have extra properties that allow cheating.

A. The Stand-Alone, Good PUF Model

In the stand-alone, good PUF model, we make the following assumptions:

1) The protocol is executed only once in a stand-alone setting, meaning that the protocol is never re-run, also not any (sub-)sessions of it. The employed PUF(s) cannot be accessed or communicated with after the end of the protocol.
2) The employed PUFs are all "good PUFs", meaning that are drawn faithfully from a previously specified distribution of PUFs and are not modified in any way afterwards, neither by malicious players nor by external adversaries. They only have the properties and functionalities expected by the honest protocol participants.

The stand-alone model is not realistic or efficiently realizable in most practical PUF-applications, but it makes a clean first scenario for studying the security of PUF-protocols.

B. The PUF Re-Use Model

In order to model the execution of multiple PUF protocols, Brzuska, Fischlin, Schröder and Katzenbeisser [1] proposed one possible method how Canetti's UC-framework [2] can be adapted to PUFs. For a detailed treatment we refer the readers to the original paper [1] and the follow-up paper by Ostrovsky, Scafuro, Visconti and Wadia [10]. We summarize the features of their model that are most relevant below.

1) It is assumed that all used PUFs are drawn faithfully from a previously specified distribution of PUFs, a so-called "PUF-family", and are not modified in any way afterwards, neither by malicious players nor by external adversaries. They only have the properties and functionalities that honest protocol participants expect from them. This feature is in common with the above stand-alone, good PUF model.
2) Only one PUF can be used per protocol session `sid`. The PUF is bound to this protocol session and cannot be used in another session.
3) The adversary does not have physical access to the PUF between the different subsessions `ssid` of a protocol.

One implicit assumption of Brzuska et al. is that the adversary cannot access the PUF between different (sub-)sessions, and that the PUF is never re-used in another protocol session. However, this assumption seems difficult to guarantee in many natural PUF applications.

To see this, consider the well-established application scenario of a PUF on a bank card, which has been issued by a central authority CA and is subsequently used in different terminals [12], [11]. To be more concrete, let us assume that

the PUF is repeatedly employed for a session key exchange between the CA and the smart-card/terminals. Since an adversary could set up fake terminals, add fake readers to the card slots of terminals, or gain temporary possession of the bank card when it is employed in different contexts (for example when the user is paying with it), a realistic assumption is that an adversary will have *repeated* temporary physical access to the PUF between the different key exchange (sub-)sessions. However, such access is not foreseen in the models and protocols of Brzuska et al.

The example illustrates that in practice, adversaries and malicious players may gain access to the PUF at least occasionally between different (sub-)sessions. This constitutes a new, relevant attack point and motivates an extension of the model of Brzuska et al. [1]. Ostrovsky et al. [10] deal with this observation in their own manner: They implicitly assume a one-time use of the PUF. However, such one-time use and subsequent destruction or locking away of the PUF results in substantial practical costs. It constitutes a theoretically acceptable, but at the same time commercially a somewhat infeasible measure.

b) The PUF Re-Use Model: We assume that at least a subset of the PUFs employed in the original protocol is used on more than one occasion, i.e., not all PUFs are used only once and destroyed immediately afterwards. The adversary or malicious parties have access to the PUF more than once, for example before, after or between different protocols or protocol (sub-)sessions (if there are any).

The description leaves some detail open, the simple reason being that many differing variants of the PUF re-use model are possible. For example, one can distinguish between the type of adversarial access: (i) full physical access, where the adversary can attempt arbitrary actions on the PUF, including arbitrary measurements or active physical modification of the PUF, or (ii) CRP access, where the adversary's actions are limited to the mere measurement of CRPs. One can also differentiate the number of occasions on which access is possible; or the relative time of the access, such as before or after the attacked protocol; or the number of CRPs the adversaries can read out during his access time. One can further distinguish between different types of re-use: Is the PUF re-used by the same parties in another instance of the same protocol, or by entirely new parties in a different protocol? Instead of declining through all possible scenarios formally here, we suggest that such differentiation should be made in the respective security analyses directly.

There is only one specific instantion we would like to define explicitly here:

c) The One-Time Posterior Access Model (PAM): In the PAM, we assume that the adversary has got access to at least a subset of all PUFs employed in the original protocol on exactly one occasion after the end of the protocol (or protocol (sub-)session, if there are any), and is furthermore limited to the measurement of standard CRPs.

We notice that the PAM is arguably the mildest possible form of the PUF re-use model. Still, it suffices to successfully attack many existing schemes as explained in [17].

C. The Bad PUF Model

The other central assumption in the good PUF model is that the players are not allowed to use "bad", fraudulent PUF-hardware with properties beyond the expected PUF functionality. This assumption can be difficult to uphold in practice.

To motivate bad PUFs, consider once more the earlier smart-card example. Let us assume that the CA issues the card that carries the PUF, and that the CA and the smart-card/terminals want to run an OT protocol in this setting. We must assume that the CA is not fully trusted by the smart-card/terminals (note that if the CA was fully trusted, then the smart-card/terminals would not require an OT implementation). However, a malicious CA can cheat easily in this scenario by putting a malicious PUF-hardware (a "bad PUF") instead of a normal PUF on the smart card. To name one example, the CA could replace the normal PUF by a pseudo random function (PRF) or a pseudo-random number generator (PRNG) with a seed s known to the CA. If the PRF will have the same, digital input-output interface as the normal PUF, such a step will remain unnoticed. Still, it enables the CA to simulate and predict all responses of this "bad PUF" without being in physical possession of it, and to break one of the essential security features of the purported "PUF" on the bankcard, namely its unpredictability. Under the assumption that the CA replaces the PUF by a PRF with a seed known to the CA, the well-known OT protocols of Rührmair [14] and Brzuska et al. [1] are no longer secure.

Abstracting from this specific example, the general problem is that in a typical two-party protocol, one of the parties can fabricate the PUF, while the other party may only use the PUF "from the outside" via a (digital) challenge-response interface. It is hard to verify that there is no unexpected, malicious functionality on the other side of the interface. From a practical perspective, this observation is most severe for electrical Strong PUFs, which are the most widely distributed Strong PUFs today. But it also holds for *integrated* optical PUFs as given by Tuyls and Skoric [23].

This motivates a systematic study of bad PUF attacks. Generally, we denote by the term *bad PUF* a hardware system that looks like a proper PUF from the outside, exhibiting a input-output behavior indistinguishable from a proper PUF, but which possesses secret, additional properties that allow cheating. Its assumed similar input-output behavior shall make it infeasible to distinguish a bad PUF from a proper PUF by digital challenge-response measurements. In order to detect bad PUFs, honest parties would need to physically open the PUF-hardware and to inspect it thoroughly, as a regular and dedicated step of the protocol. While detection of bad PUFs would not even be guaranteed by such a step (adversaries would presumably develop obfuscation techniques), it would surely destroy the opened PUF, even if it was non-manipulated. In addition, the inspection step would be beyond the capabilities of an average user.

This makes bad PUFs a very simple and effective way to cheat. From an abstract perspective, bad PUFs exploit the fact that PUFs are real physical objects. Unlike the clean binary strings exchanged in classical cryptographic protocols, these objects may bring about unwanted properties. They can act as real, physical "Trojans" and other malicious hardware.

Even though there is a practically infinite number of possibilities how Strong PUFs can act, two types of bad PUFs that we focus on are (i) PUFs that are numerically simulatable by their manufacturer (but by no one else), and (ii) bad PUFs that "log" or record all challenges that have been applied to them. Both are particularly easy to implement, but suffice for attacks on existing protocols as demonstrated in [17].

d) Simulatable Bad PUFs (SIM-PUFs): A simulatable PUF (or SIM-PUF, for short) is a hardware system that looks like a PUF, having a challenge-response interface etc., but which possesses a simulation algorithm Sim. Sim takes as input any challenge c, and computes in polynomial time the corresponding response r. It is assumed that Sim has been derived during the fabrication of the simulatable PUF via the special construction of the PUF. External parties who merely have access to the simulatable PUF after fabrication are not able to derive a simulation model.

In practice there are several possibilities for implementing simulatable PUFs. A straightforward and very efficient way is to use a trapdoor one-way permutation or pseudo random function g_s based on a short digital seed s. The hardware of the simulatable PUF simply implements g_s. Whenever the PUF is interrogated over the digital interface with a challenge c, the hardware outputs the response $r = g_s(c)$.

The party who manufactured the PUF knows both g as well as seed s and can easily simulate the input-output behavior of the PUF. Furthermore, if a cryptographically hard pseudo-random function is used, it is practically infeasible for the honest parties to distinguish the bad PUF from a proper PUF with a real, random output.

e) Challenge-Logging Bad PUFs (CL-PUFs): A second feature that bad PUFs may possess is challenge-logging. A challenge-logging PUF (CL-PUF for short) with secret challenge c^*, also called the access challenge, is a malicious piece of hardware that looks like a proper PUF from the outside (with a challenge-response interface etc.), but which possesses the following properties:

1) Except for one input challenge c^*, the challenge-response behavior of a CL-PUF is exactly like that of an underlying, "normal" PUF. Whenever a challenge c unequal to c^* is applied to the CL-PUF via its interface, the challenge is passed on to the underlying PUF. The corresponding response r is obtained from the latter, and the CL-PUF uses this response r as its output.
2) The CL-PUF has a non-volatile memory (NVM) module in which it automatically records all challenges that have been applied to it.
3) When challenge c^* is applied to the CL-PUF, it does not pass on this challenge to the underlying PUF as usual. Instead, the CL-PUF outputs the entire content of the non-volatile memory module (i.e., all challenges that have previously been applied to it) via the challenge-response interface, and erases the content of the NVM module.

If the PUF has a large, preferably exponential challenge set, then the probability that someone by chance inputs c^* and detects the challenge-logging feature is negligibly small. Please note that many alternative ways for activating the output mode of the challenge-logger are conceivable, such as radiowave triggering etc., and even entirely other forms of logging and read-out "modes" of the logger are possible (see below).

Finally, we observe that there are two fundamentally different types of CL-PUFs: PUFs that have been malicously constructed with a challenge-logger from the start; and CL-PUFs where a logger-module has been added externally by malicious parties after their construction. The former seem yet more easy to implement, but also the second type is a viable attack strategy. In any way, CL-PUFs act as real, physical Trojans: They record and store security-critical information and pass it on to the adversary when he holds possession of the PUF again.

f) Advanced Bad PUFs: How "bad" can a PUF be? Having so far focused on simple features, which still suffice to attack many existing protocols, we now mention one particularly "super-bad" PUF: A *Communicating PUF* transmits the challenge, the response, or both, to fraudulent parties. The transmission could be carried out in real time, or may be delayed to later, when the PUF is released from the control of the honest parties. It is relatively straightforward that such a feature destroys the security of all existing protocols. Necessary, but also very costly countermeasures are shielding the PUF during the protocol and destroying them immediately afterwards.

Many other examples of advanced bad PUFs are conceivable, see [17]. Actually, any such bad PUF types have to be taken into consideration when the security of a PUF protocol is analyzed. But we notice that the earlier, simpler types of SIM-PUFs and CL-PUFs already suffice for attacking many protocols.

III. Counter Measures: The Need for Erasable and Certifiable PUFs

The findings of our analysis are somewhat alarming. They suggest that attack models and protocol design for "advanced" Strong PUF protocols should be strongly reconsidered. As PUFs are hardware systems that can have hidden extra features, new strategies become necessary here.

One possible countermeasure is to (i) allow additional computational assumptions in the protocols; (ii) assume that the PUFs can be shielded during the course of the protocol in order to prevent communication between the bad PUF and malicious parties, in particular, that the PUF is no Communicating PUF; and (iii) to use each PUF only once, destroying it at the end of the protocol in order to prevent access by adversaries after the protocol. This path is taken by Ostrovsky

et al. in their work [10]. However, there are some downsides associated with this approach: The introduction of additional computational assumption takes away some of the appeal of Strong PUFs as a new, independent cryptographic primitive. The effective shielding of PUFs until their destruction is hard to achieve in concurrent, complex environments. And, perhaps most importantly, the one-time use and destruction of the used PUFs after each protocol execution is extremely costly in practice. It constitutes a theoretically viable, but practically and commercially essentially infeasible measure. They lead us to the question whether other approaches for fighting the PUF re-use model and bad PUFs exist in practice.

Two other, direct countermeasures against the PUF re-use model and bad PUFs are so-called Erasable and Certifiable PUFs.

g) Erasable PUFs: Erasable PUFs are Strong PUFs with the additional feature that single responses can be erased from the PUF (i.e., made impossible to read out forever) without affecting any of the other responses. Erasable PUFs have been considered for the first time by Rührmair, Algasinger and Jaeger in [18], who also suggest an implementation based on so-called crossbar structures. This implementation is very area consuming, however. Area efficient implementations have not been suggested up to this date. In order to better understand the challenges and the novelty behind Erasable PUF design, consider two of the currently most established Strong PUF designs: Arbiter PUFs [21] and optical PUFs [12]. In both designs, many subparts of the PUF interact in order to generate a response. If one response shall be altered or erased, at least one of the subparts must be changed. In the example of optical PUFs, certain subparts of the scattering platelet would need to be modified; in the case of the Arbiter PUF, at least one internal delay value would need to be altered. But this will necessarily also affect and modify other responses, contradicting the requirements of an Erasable PUF. Reconfigurable PUFs [9] are unsuited as Erasable PUFs for the same reason: Their reconfiguration operation by definition alters all responses of the PUF in one step. This makes any previously collected CRPs of the PUF invalid.

If the area efficient, direct implementation of Erasable PUFs remains difficult in the future, then an alternative strategy could be equipping Strong PUFs with a surrounding control logic. This logic is supposed to guard and regulate the access to the Strong PUF's challenge-response interface; such constructions are also known as Controlled PUFs [3]. Along these lines, one could construct *Logically Erasable PUFs* by letting the control logic maintain some record of the previously applied and of the erased challenges (e.g., in the form of an authenticated hash tree). Also Logically Reconfigurable PUFs (LR-PUFs) as introduced by Katzenbeisser et al. [6] can be an option in this context. They allow the manufacturer of the PUF to collect a CRP-list that remains valid even after many reconfiguration operations. This may suffice to ensure the security of certain protocols in the PUF re-use model. We remark, however, that such versions of Controlled PUFs introduce additional assumptions, for example that it is impossible to circumvent, modify or tamper the control logic around the underlying Strong PUF.

h) Certifiable PUFs: A straightforward countermeasure against bad PUFs seems to "authenticate" or "certify" the PUF in one way or the other in order to detect bad PUFs. For example, a trusted authority (TA) could send a list of CRPs as a "fingerprint" of a genuine PUF to the players before any protocol execution. On closer inspection, however, this countermeasure turns out to be very problematic, and pretty much falls apart.

First of all, the use of a TA that needs to be called in every single protocol session would make the use of PUFs in security protocols obsolete. The aspired functionalities could then be implemented in a much simpler fashion directly via the TA, avoiding the significant effort of physically transferring a PUF during the protocol. Secondly, CRP-based authentication does not rule out externally added malicious hardware, such as external challenge loggers. The latter do not affect the CRP-behavior of an existing (and previously certified) PUF.

Meaningful "certification" of a PUF hence requires not only to "identify" a PUF. It also must (i) exclude that external parts have been added to the PUF or that the PUF-hardware has been manipulated; and (ii) it should work offline, i.e., it must avoid calling a central TA in every execution of the protocol. Currently, no protocols or PUF implementations that realize these two properties have been considered in the literature. Given the current state of the field, it seems hard to design such methods, even more so at low costs. As discussed earlier, physical inspection of the inner configuration of the PUF as a regular protocol step seems no viable possibility. Furthermore, if efficient methods for certifying the integrity of (PUF-)hardware existed, then the same methods could be applied to protect security modules built on classical keys, making PUFs obsolete.

Summarizing, Certifiable PUFs are PUFs that allow an offline certification of the fact that they have only those properties that the honest parties expect from them. It is possible to verify that they have been drawn faithfully from the expected PUF distribution, and that they have not been modified by anyone in any way afterwards. Currently, however, no measures whatsoever have been considered in the literature how such authentication can be achieved.

The combination of certifiability and erasability (or variants such as logical erasability/reconfigurability) in a single piece of hardware therefore poses a highly relevant, but very challenging open problem to the PUF hardware community and security community at large. It should be resolved in order to restore the full applicability of Strong PUFs as a general, broadly, and efficiently usable cryptographic tool. It would allow PUF protocols in complex environments without additional computational assumptions, and without an economically unrealistic one-time use of PUFs.

IV. Summary

We surveyed a number of attack models for Strong PUF protocols, including the *"PUF re-use model"* and the *"bad*

PUF model". These models constitute practically relevant and hard-to-detect attack strategies, and are strongly relevant for practical PUF usage scenarios. We explained countermeasures, most notably, the concept of Erasable PUFs and Certifiable PUFs. Their efficient realization poses an open and challenging research problem for the PUF hardware community.

REFERENCES

[1] C. Brzuska, M. Fischlin, H. Schröder, and S. Katzenbeisser: *Physical Unclonable Functions in the Universal Composition Framework.* CRYPTO 2011. Full version in Cryptology ePrint Archive, Report 2011/681, 2011.

[2] R. Canetti: *Universally Composable Security: A New Paradigm for Cryptographic Protocols*. FOCS 2001: 136-145.

[3] B. Gassend, M. van Dijk, D.E. Clarke, E. Torlak, S. Devadas, and P. Tuyls: *Controlled physical random functions and applications.* ACM TISSEC 10(4), 2008.

[4] B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas: *Silicon physical random functions*. ACM CCS 2002.

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

[6] S. Katzenbeisser, C. Koabas, V. van der Leest, A.-R. Sadeghi, G. J. Schrijen, and C. Wachsmann: *Recyclable PUFs: Logically Reconfigurable PUFs*. Journal of Cryptographic Engineering 1(3): 177-186 (2011).

[7] J. Kilian: *Founding cryptography on oblivious transfer*. STOC, 1988.

[8] S. S. Kumar, J. Guajardo, R. Maes, G. J. Schrijen, and P. Tuyls: *The Butterfly PUF: Protecting IP on every FPGA*. HOST 2008.

[9] K. Kursawe, A. R. Sadeghi, D. Schellekens, P. Tuyls, and B. Skoric: *Reconfigurable physical unclonable functions – Enabling technology for tamper-resistant storage*. HOST 2009.

[10] R. Ostrovsky, A. Scafuro, I. Visconti, and A. Wadia: *Universally Composable Secure Computation with (Malicious) Physically Uncloneable Functions*. Eurocrypt 2013.

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

[12] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld: *Physical One-Way Functions*, Science, vol. 297, 2002.

[13] R. Rivest: *Illegitimi non carborundum*. Invited keynote talk, CRYPTO 2011.

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

[15] U. Rührmair, H. Busch, and S. Katzenbeisser: *Strong PUFs: Models, Constructions and Security Proofs.* In A.-R. Sadeghi, P. Tuyls (Editors): Towards Hardware Intrinsic Security: Foundation and Practice. Springer, 2010.

[16] U. Rührmair and M. van Dijk: *Practical Security Analysis of PUF-based Two-Player Protocols*. CHES 2012.

[17] U. Rührmair and M. van Dijk: *PUFs in Security Protocols: Attack Models and Security Evaluations*. IEEE S&P 2013.

[18] U. Rührmair, C. Jaeger, and M. Algasinger: *An Attack on PUF-based Session Key Exchange and a Hardware-based Countermeasure: Erasable PUFs*. Financial Cryptography, 2011.

[19] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber: *Modeling Attacks on Physical Unclonable Functions*. ACM CCS, 2010.

[20] U. Rührmair, J. Sölter, and F. Sehnke: *On the Foundations of Physical Unclonable Functions.* Cryptology ePrint Archive, Report 2009/277, 2009.

[21] G. E. Suh and S. Devadas: *Physical Unclonable Functions for Device Authentication and Secret Key Generation.* DAC 2007.

[22] P. Tuyls, G. J. Schrijen, B. Skoric, J. van Geloven, N. Verhaegh, and R. Wolters *Read-Proof Hardware from Protective Coatings.* CHES 2006.

[23] P. Tuyls and B. Skoric: *Strong Authentication with Physical Unclonable Functions.* In: Security, Privacy and Trust in Modern Data Management, 2007.