On the Formal Foundations of PUFs and Related Primitives

vorgelegt von
Ulrich Rührmair, MSc.,
geb. in München

von der Fakultät IV — Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
— Dr. rer. nat. —

genehmigte Dissertation

Promotionsausschuss:

Vorsitzende: Prof. Anja Feldmann, PhD
Gutachter: Prof. Dr. Jean-Pierre Seifert
Gutachter: Prof. Marten van Dijk, PhD
Gutachter: Prof. Srinivas Devadas, PhD
Gutachter: Prof. Dr. Marc Fischlin

Tag der wissenschaftlichen Aussprache: 30. Juni 2016

Berlin 2016

On the Formal Foundations of PUFs and Related Primitives

Ulrich Rührmair

October 19, 2016

To my parents
and family

Like the legend of the phoenix
All ends with beginnings
What keeps the planet spinning
The force from the beginning

We've come too far to give up who we are
So let's raise the bar and our cups to the stars

She's up all night to the sun
I'm up all night to get some
She's up all night for good fun
I'm up all night to get lucky

We're up all night to the sun
We're up all night to get some
We're up all night for good fun
We're up all night to get lucky

We're up all night to get lucky
We're up all night to get lucky
We're up all night to get lucky
We're up all night to get lucky

Repeat and fade...

Get Lucky
DAFT PUNK FEAT. PHARELL
WILLIAMS & NILE RODGERS

Contents

Part I

Introduction and Overview

Chapter 1

Primer and Formalities

1.1 Motivation and Contextualization

In the year 2005, the ACM president at the time, David A. Patterson, proposed within the Communications of the ACM the so-called *"SPUR manifesto"* [43]. Its aim was to guide the computer and communication (C&C) research at the brink from the 20th to the 21st century. Together with usability and reliability, he identified security&privacy as one of three major topics in future C&C studies, stating that

> *"In my view, we have taken ideas from the 1970s and 1980s to their logical extreme, providing remarkably fast and cheap C&C to hundreds of millions of people. But we now are all painfully aware of the drawbacks of 20th century C&C. [...] We must protect the security and privacy of C&C users from criminals and terrorists while preventing the Orwellian vision of Big Brother. C&C in the 21st century should be as safe as 20th century banking."*

Around the same time, in a completely independently vein of activities, a novel approach in cryptography and security began to gain popularity under the name of a *"physical unclonable function"* or *"PUF"*, for short [41, 42, 22]. In a nutshell, a PUF is a (partly) disordered physical system, which exhibits some form of *"input-output"* or *"challenge-response"* functionality. Each response of the PUF to an applied challenge thereby is a function of both the unique physical disorder in the PUF and of the challenge itself.

Due to their randomly disordered small-scale structure, PUFs cannot be physically copied or cloned exactly in practice, even if their entire structure is known to the adversary. This is in stark contrast to a classical cryptographic key, or to the architecture of a standard digital circuit, which can both be duplicated easily once they have become known. Their hybrid nature between physics and mathematics allows qualitatively new security applications, for example identification schemes that operate without digital, permanent keys in the hardware of the prover.

Interestingly, PUFs exactly follow the paradigms of the SPUR manifesto, as cited above: They extend the digital and logical concepts of standard security techniques, which are based on digital keys and mathematical assumptions. They instead explore new approaches and assumptions in the physical domain. And, finally, they bear the promise of not just delivering fast and cheap C&C, but also inexpensive, lightweight security to millions of users.

For these and other reasons, PUFs have undergone a tremendous development within the scientific community in the last decade (compare [52, 53]), and have become one of the most prospering subfields of cryptography and hardware security in general. Instead of giving a comprehensive history of the field, which has been done elsewhere [53], let us merely illustrate this rapid development by a few stunning, exemplary facts [52]: The two root papers of the area by Pappu et al. [42] and Gassend et al. [22] to date have been quoted around 800 times according to Google scholar. PUF-papers have appeared at every flagship venue of the wider cryptography and security community, including EUROCRYPT [40], CRYPTO [6] ASIACRYPT [1, 17], ACM CCS [72, 83], IEEE S&P [2, 60], the Journal of Cryptology [37], IEEE T-IFS [11, 36, 76], and ACM TISSEC [23]. CHES and HOST, the two major hardware security conferences, have continuously hosted one or even two dedicated PUF sessions in each year since 2010 [79, 80]. Also the circuit design community has become strongly engaged in the topic; for example, one of its premium events DATE achieved an all-time record of eleven dedicated PUF papers in one single conference in 2014 [81].

Given this remarkable expansion, however, the foundations of the field have increasingly moved into focus. Especially in the early days, many PUF works solely focused on implementational and hardware aspects. Different PUF types were often not explicitly distinguished, and the security properties required in a certain application were not worked out clearly. This sometimes made it hard to judge new implementations or applications for their viability. Furthermore, it made communication between the different disciplines involved in PUF-research unnecessarily difficult; sometimes it even prevented a sound and timely proliferation of the seminal ideas that underlie PUF-research into the neighboring communities.

For a healthy long-term development, it therefore seems both pressing and indispensible to lay the area's theoretical foundations solidly. This necessity has already been subsumed in the context of classical cryptography in a picturesque manner by Oded Goldreich, who quotes his father in saying that

> *"It is possible to build a cabin with no foundations, but not a lasting building."* [25]

Following this credo, our thesis is wholeheartedly devoted to the foundations of PUFs and related disorder-based primitives within the area of so-called *"physical cryptography"* [66, 58]. Among others, we investigate the following rich and diverse topics: Formal definitions and classifications of PUFs; security proofs for disorder-based primitives such as UNOs; novel Turing machine models that can incorporate physical actions of the adversary; the potential of PUFs in advanced cryptographic and security protocols; and the effective practical security of PUF-schemes under realistic attack models.

We stress, however, that our thesis does not solely focus on isolated or overly theoretical results. Rather, it places great emphasis on the correct translation of theoretical findings into “real” application scenarios, revealing a number of shortcomings and vulnerabilities of previous works in this direction. Overall, we sincerely hope to have sustainably advanced the foundations of the area, contributing to a sound basis for its long-term development.

In the context of the overall thesis, this first chapter thereby mainly provides a brief overview and settles some necessary formalities. Section 1.1 gives a short motivation and contextualization: Section 1.2 subsumes our original contributions; Section 1.3 comprehensively lists all publications that are used as chapters of this thesis, and also details the author’s individual contributions to these publications, as required by the dissertation statutes of the TU Berlin; and Section 1.4 provides a short outline and a guide to the reader. The main technical contributions are then being made afterwards, namely in Chapters 2 to 8. Readers mostly interested in the latter are encouraged to jump directly to these chapters, perhaps after reading the guidelines in Section 1.4.

1.2 Original Contributions of this Thesis

Being a cumulative thesis, this work contains as its chapters several publications of the candidate in their original, published form. These papers make (or have made at the time of their appearance) the following original contributions:

- We investigate the formal foundations and correct classification of PUFs and related primitives. The paper presented in Chapter 7 [74] was the historically first dedicated publication on this topic in 2009. It formally specifies the concepts of a “Strong PUF” and of an “Obfuscating PUF” or “Weak PUF”, using the ever first game-theoretic definitions of PUF security. Among other things, the chapter also investigates the maximal information content or entropy in a PUF, and discusses formal issues with asymptotic PUF definitions that existed at the time of its publication. The core of the paper goes back to an invited talk of the candidate at a workshop in Dagstuhl in 2008 [45]. This part of our work arguably inspired a number of follow-up activities on the foundations of PUFs, including works at major conferences like IEEE S&P [2]. It is the most often quoted paper of this thesis, with around 70 quotations at the time of the submission of this thesis in 2014, and over 100 citations today (status: September 26, 2016). The survey paper presented in Chapter 2 follows the same route and classifies and differentiates several disorder-based primitives.

- We explicitly consider Strong PUFs as an advanced cryptographic primitive (as opposed to a mere security tool for key storage). Along these lines, we are the first to actually realize PUF-based oblivious transfer in 2010 [47], showing the strong potential along these lines. The corresponding paper is given in Chapter 3. Via a number of well-known reductions, this proves that Strong PUFs can, in principle, implement any secure multi-party computation, key exchange, or bit commitment protocols. As above, this work arguably inspired a large number of follow-up works at major conferences, including CRYPTO 2011 [6] and EUROCRYPT 2013 [40].

- We investigate the concrete security of advanced PUF protocols in realistic application scenarios, discovering attacks on several, previously secure schemes. We start in Chapter 4 by an analysis of oblivious transfer (OT) and bit commitment (BC) protocols by Brzuska et al. from CYPTO 2011 [6], and show the existence of quadratic attacks on these protocols. While such attacks usually are considered a nuisance in asymptotic frameworks, we illustrate that they do have an effect in the context of concrete PUF implementations: They make the OT and BC protocols insecure if they are implemented via optical PUFs, as suggested explicitly in the original paper at CRYPTO 2011 [6]. This means that the protocols cannot be used securely in practice in the way suggested in the original paper [6]. The paper underlying Chapter 4 was published at CHES 2012, and was voted one of the best papers of the conference; it was subsequently invited for submission at the Journal of Cryptographic Engineering.

- We continue by analyzing a PUF-scheme by Tuyls and Skoric from 2007 [86] in Chapter 5. It employs a PUF on a bank card to exchange a session key between the bank on the one hand and the card (or a terminal where the card is inserted) on the other hand. We show, however, that adversaries with multiple access to the PUF on the card can derive previous session keys. Since bank cards are usually employed multiple times in various terminals, this appears as a very natural and realistic attack scenario.

- Partly generalizing from the concrete attack on the scheme by Tuyls and Skoric, we semi-formally specify two general new adversarial models for Strong PUF protocols: The so-called *"bad PUF model"* and the *"PUF re-use model"* [18, 60] in Chapter 6. [1] We show that a large number of existing, advanced PUF protocols are not secure in these attack models, including the schemes for OT, BC and key exchange by Brzuska et al. from CRYPTO 2011 [6], and partly also from Ostrovsky et al. from EUROCRYPT 2013 [40]. It must be stressed that our attack models lie outside the original security models of these papers, but that they appear highly realistic, and that PUF protocols would be faced with them when used in practice. The paper underlying Chapter 6 was published at the IEEE Symposium on Security and Privacy, the top conference in the entire area of computer security and cryptography.

- As countermeasures to the bad PUF model and the PUF re-use model, we introduce the novel PUF-concepts of a *"Certifiable PUF"* and an *"Erasable PUF"* in Chapters 5 and 6. They could restore the security of advanced PUF protocols in practical settings.

  We also make the first steps towards the hardware implementation of Erasable PUFs via so-called crossbar nano-structures and ALILE diodes [64] in Chapter 5, including proof-of-concept fabrication and measurement of such diodes. They

[1]We remark that one of these two models, namely the bad PUF model has been discussed completely independently and around the same time as the work of the candidate by Ostrovsky et al. under the term "malicious PUFs" (compare [18] and [39]). The PUF re-use model, however, is solely the work of the candidate, and has not been touched upon by other authors.

show the large variations in the diodes, and prove the possibility for erasing their information content by applying a certain threshold voltage.

- In Chapter 8, we finally develop a new Turing machine model that can explicitly deal with physical objects, so-called *"physical Turing machines"*. The novel model allows us lead the first formal security proof in the PUF area that explicitly and directly includes *physical* features, such as the physical unclonability of PUFs and related primitives. The proof also was the first formal proof on so-called *"unique objects (UNOs)"* (see Chapters 2 and 8), and the first *asymptotic* security proof in the PUF area at the time of its publication in 2011. [2]

1.3 Relevant Publications and Individual Contributions of the Candidate

In compliance with the doctoral statutes of the TU Berlin, we below list the seven scientific publications of the candidate that are used as the chapters of this cumulative thesis, and explicitly describe the candidate's individual contributions to each single publication.

According to Google scholar [26], these publications altogether have been quoted around 330 times by September 26, 2016, with the candidate being the first author or sole author of all of them. We also provide the latest citation numbers of each single paper for the same cutoff date below, again according to Google scholar [26].

- CHAPTER 1 has been solely written by the candidate.

- CHAPTER 2 uses the publication

  - U. Rührmair, S. Devadas, F. Koushanfar: *Security based on Physical Unclonability and Disorder*. In: Introduction to Hardware Security and Trust, M. Tehranipoor and C. Wang (Eds.), pp. 65-102, Springer, 2012,

  which is a book chapter with Sections 4.1 – 4.7. The candidate's contributions are as follows: He initated and conceptualized the publication, and mainly devised its structure, including the classification of disorder-based primitives into unique objects, Weak PUFs, Strong PUFs, and Controlled PUFs. He wrote the largest parts of Sections 4.1, 4.2, and 4.4, and substantial fractions of Sections 4.6. Parts of Sections 4.3 were contributed by him, too.

  According to Google scholar, this work has been cited 52 times to this date [26].

[2] We would like to mention for completeness that the manuscript in fact dates back much further to 2006, having been distributed to a number of colleagues at the time, including Stefan Katzenbeisser, Sven Kosub and Helmut Veith, but that it was only put online in 2011.

- CHAPTER 3 employs the publication

  - U. Rührmair: *Oblivious Transfer Based on Physical Unclonable Functions*. TRUST 2010, Lecture Notes in Computer Science, Vol. 6101, pp. 430-440, Springer, 2010.

  It is a single author paper of the candidate.

  According to Google scholar, it has been cited 44 times to this date [26].

- CHAPTER 4 utilizes the publication

  - U. Rührmair, M. van Dijk: *Practical Security Analysis of PUF-based Two-Player Protocols*. Cryptographic Hardware and Embedded Systems (CHES 2012), Lecture Notes in Computer Science, Vol. 7428, pp. 251-267, Springer, 2012.

  The candidate initialized and conceptualized this paper. Apart from Lemma 3 in Section 3, the publication is essentially due to him. He discovered the quadratic attack, developed the interactive hashing based alternative protocols, and conducted all necessary security analyses and most of the writing.

  According to Google scholar, it has been cited 23 times to this date [26]. It was selected one of the best papers at CHES 2012 and invited for submission to the Journal of Cryptographic Engineering.

  This created the following, related publication, which is explicitly not used in this cumulative thesis:

  - U. Rührmair, M. van Dijk: *On the practical use of physical unclonable functions in oblivious transfer and bit commitment protocols*. Journal of Cryptographic Engineering 3(1), pp. 17-28, 2013.

- CHAPTER 5 consists of the publication

  - U. Rührmair, C. Jaeger, M. Algasinger: *An Attack on PUF-Based Session Key Exchange and a Hardware-Based Countermeasure: Erasable PUFs*. Financial Cryptography and Data Securiy (FC 2011), Lecture Notes in Computer Science, Vol. 7035, pp. 190-204, Springer, 2012.

  As above, the candidate initialized and conceptualized the paper. The attack on the session key exchange protocol is due to him, and so is the concept of an *"Erasable PUF"* put forward in the paper. Sections 1, 2, 3, 4, and 7 were written by the candidate. The physical experiments reported in Section 5 and 6 are due to his co-authors, while the writing of Sections 5 and 6 has mainly been accomplished by the candidate.

  According to Google scholar, it has been cited 29 times to this date [26].

- CHAPTER 6 uses the publication

  - U. Rührmair, M. van Dijk: *PUFs in Security Protocols: Attack Models and Security Evaluations*. IEEE Symposium on Security and Privacy 2013, pp. 286-300, 2013.

  As above, this paper has been initiated and led by the candidate. The described attack models and presented attacks are due to him. Apart from parts of Section III.E, the paper has been written by the candidate.

  This work was published at the premium venue in cryptography and security, the IEEE Symposium on Security and Privacy, with an acceptance rate of only 12% in 2013. According to Google scholar, it has been cited 52 times to this date [26].

  Two related existing papers, which explicitly have not been used in this cumulative thesis, are

  - M. van Dijk, U. Rührmair: *Physical Unclonable Functions in Cryptographic Protocols: Security Proofs and Impossibility Results.* IACR Cryptology ePrint Archive, Report 2012/228, 2012.
  - M. van Dijk, U. Rührmair: *Protocol attacks on advanced PUF protocols and countermeasures*. Design, Automation and Test in Europe (DATE 2014), pp. 1-6, 2014.

- CHAPTER 7 utilizes the publication

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

  Again, the paper was initiated, conceptualized and led by the candidate. Apart from the machine learning results in the appendix, the main contributions and writing are due to him, including the differentiation and classifications of PUF variants, the adversarial models and PUF-definitions, the information-theoretic analysis of the maximal entropy in a PUF, and the discussion of previous PUF definitions.

  According to Google scholar, it has been cited 102 times to this date [26]. This makes it the most quoted work of this thesis.

- CHAPTER 8 uses the publication

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

  It is a single author paper by the candidate.

  According to Google scholar, it has been cited 6 times to this date [26].

We explicitly remark that the above publications do not constitute the complete publication list of the candidate. A number of other works has been used for a second thesis entitled *"Disorder-based Security Hardware"*, which has been submitted to the TU München for the degree of a *"Dr.-Ing."*. That second thesis concentrates on hardware-related and implementational work, and employs a fully disjoint set of publications and different scientific methodologies. It has been clearly demarcated from this thesis, which focuses on theoretical and foundational aspects.

1.4 Guide to the Reader and Thesis Outline

We conclude this introductory chapter by a compact overview of the thesis' organization, including some guidelines and hints for readers.

PART I of the thesis, which contains Chapters 1 and 2, gives an introduction to and overview of the topic:

- We start in this Chapter 1 by a brief motivation and contextualization of our work, clarifying a number of technacalities and formalities.
- In Chapter 2, we then present a more detailed introduction to the area, specifying various PUF-like primitives like physically obfuscated keys (POKs) or Weak PUFs, Strong PUFs, and unique objects (UNOs). Readers getting in touch with PUFs for the first time might want to at least skim through Chapter 2 in order to prepare them for the more advanced topics in later chapters.

PART II, which consists of Chapters 3 to 6, then investigates the fundamental usability of Strong PUFs in cryptographic protocols:

- To start with, Chapter 3 shows that in theory oblivious transfer (OT) can be realized by PUFs — at least in stand-alone settings and under certain assumptions on the used PUF, which must be secure and non-manipulated (compare Chapter 6). This reveals a previously unexpected power of Strong PUFs as a cryptographic tool, as a very large number of other cryptographic protocols can be based on OT [29].
- Chapter 4 subsequently analyzes the practical security of PUF protocols for OT and bit commitment (BC) that have been proposed at Crypto 2011 [6]. We show that these protocols are not secure if optical PUFs are employed, as explicitly suggested in [6]. Our interactive hashing based OT-protocol of Chapter 3 [47] comparably has got better security.
- Chapter 5 continues our practical security analyses, and investigates a session key exchange protocol by Tuyls and Skoric from 2007 [86]. The protocol has been suggested for key exchange between a bank card/terminal/PUF on the one hand and the bank headquarters on the other hand. We are able to show that it is vulnerable if an adversary gets access to the PUF on the bank card mutiple times.

- Chapter 6 subsequently generalizes our earlier observations, concluding the strand of work on PUF protocols in the second part of this thesis. We first formally introduce two formal, general adversarial models, the so-called *"PUF re-use model"* and the *"bad PUF model"*, which are close to practical Strong PUF usage scenarios. We then demonstrate vulnerabilities of a considerable number of recent Strong PUF schemes in the two models, including protocols published by Brzuska et al. at Crypto 2011 [6], and partly also by Ostrovsky et al. from Eurocrypt 2013 [40]. Both Chapters 5 and 6 suggest countermeasures against these alarming vulnerabilities, introducing the concepts of *"Erasable PUFs"* and *"Certifiable PUFs"*. Efficient implementation of these new types of PUFs would restore the usability of Strong PUFs as a universal tool in advanced cryptographic protocols. Chapter 5 reports first steps towards the physical implementation of Erasable PUFs by nanoscale crossbar architectures.

PART III of the thesis, which comprises of Chapters 7 and 8, then deals with the mathematical formalization of PUFs and related pritimitives:

- Chapter 7 takes a dedicated look at the foundations of PUFs, and classifies different PUF types with respect to their security features. The chapter also investigates the soundness of several earlier PUF definitions, and points to a number of problems.
- Chapter 8 continues this line of investigations, refining it yet further: We introduce a new Turing machine model, so-called physical Turing machines, and lead the very first formal security proof based on this new model. One explicit advantage of physical Turing machines is that they allow full asymptotic treatment of PUFs and related primitives like UNOs. Another upside is that they allow a direct formalization of the physical unclonability of PUFs and UNOs, in opposition to all earlier PUF definitions.

PART IV, finally, concludes the thesis. It summarizes its main content in condensed form, and provides an outlook on promising future research topics in the field.

We remark that wherever possible, the thesis has beeen written in a modular fashion, enabling readers to easily jump between different parts. To make this yet simpler, every chapter begins with a short description and contextualization of the following publication, contextualizing its role within the entire thesis. Further, all included articles are self-contained by their very nature. The resulting modular accessibility could even be seen as a general advantage of cumulative theses.

Finally: Enjoy reading!

Chapter 2

Introduction to PUFs and Related Primitives, Or: Security Based on Physical Unclonability and Disorder

After the general overview of Chapter 1, this second chapter finally starts the sequence of technical contributions. It surveys the area of PUFs and related primitives, all of which utilize the phenomena of physical disorder and physical unclonability in one way or the other for security purposes. The actual primitives that we discuss include so-called unique objects (UNOs), Weak PUFs, Strong PUFs, Public PUFs, and SIMPL systems. We survey and categorize the existing literature, provide brief definitions and specifications of said primitives, discuss their possible implementation via optical, silicon or radio-frequency techniques, and present common application scenarios. We also give a detailed outlook on future research opportunities. In doing so, the chapter explicitly and significantly contributes to the foundations of the area, which are our unifying theme in this thesis.

The chapter is particularly targeted for readers who are getting in touch with the PUF-area and its underlying concepts for the first time. It equips them with the necessary background for the upcoming parts of this thesis. The concrete paper that we employ is

- U. Rührmair, S. Devadas, F. Koushanfar: *Security based on Physical Unclonability and Disorder*. In: Introduction to Hardware Security and Trust. M. Tehranipoor and C. Wang (Eds.), pp. 65-102, Springer, 2012.

According to Google scholar, it has been cited 73 times to this date [26].

The candidate would like to express his gratitude to have one of the founding fathers of the field as a co-author in this publication.

Security based on Physical Unclonability and Disorder

Ulrich Rührmair[†,1], Srinivas Devadas[2], Farinaz Koushanfar[†,3]

[†]These two authors have equally contributed to this book chapter

[1]Computer Science, Technische Universität München
[2]Electrical Engineering and Computer Science, Massachusetts Institute of Technology
[3]Electrical and Computer Engineering, Rice University

Abstract. Identification, authentication, and integrity checking are important tasks for ensuring the security and protection of valuable objects, devices, programs, and data. The utilization of the microscopic, random and unclonable disorder of physical media for such security tasks has recently gained increasing attention. Wherever applicable, the harnessing of disorder can lead to intriguing advantages: First, it can avoid the permanent storage of digital secret keys in vulnerable hardware, promising to make the resulting systems more resilient against invasive and malware attacks. Second, random physical disorder has the natural feature of being very hard to clone and to forge: Fully controlling the micro- and nanoscale fabrication variations in physical media is extremely difficult and, even if possible, prohibitively expensive. Third, utilization of the natural disorder and entropy in physical systems can sometimes enable cryptographic protocols whose security does not rest on the usual unproven number-theoretic assumptions like factoring and discrete log, creating an alternate foundation for cryptography. Physical Unclonable Functions or PUFs are perhaps the best known representative of this new class of "disordered" cryptoprimitives, but there are also others. In this chapter, we provide a classification for past and ongoing work in physical disorder based security alongside with security analyses and implementation examples. We will also outline some open problems and future research opportunities in the area.

1 Introduction

Since the number of networked smart objects, programs, and data is constantly increasing, there is an equally growing demand to ensure the security and reliability of these units. Since they are pervasive in our daily lives, this issue has become a significant societal challenge. One central task lies in realizing secure and reliable identification, authentication, and integrity checking of these systems.

Traditional security methods based on secret digital keys often do not provide adequate solutions for this purpose. One major point of vulnerability relates to their hardware implementations and key storage: A whole host of attacks for extracting, estimating, or cloning secret keys that are stored digitally in non-volatile memory have been developed and reported over the past several years. The situation is especially problematic for embedded and mobile low power devices with a small form factor, where the adversaries can often gain full and direct access to the device. For many FPGA-based reconfigurable devices, which are increasingly growing in market share, the permanent storage of secret keys can be a problem: Integrating secure non-volatile memory (NVM) on FPGAs incurs additional costs and fabrication overhead and, thus, it is often not included. Therefore, keys have to either be stored in external memory, where they are highly vulnerable, or an additional back-up battery to power on-chip volatile storage must be used, which increases cost and system complexity. We refer interested readers to Chapter 6 of this book for a full discussion of FPGA vulnerabilities and security.

Over recent years, an alternative security approach has therefore emerged, which is based on the inherent, hard-to-forge and unique disorder of physical objects. It constitutes a promising alternative which can address the standing challenges of classical security that were described above. Two major classes of disorder-based security systems that have been proposed are *Unique Objects (UNOs)* and *Physical Unclonable Functions (PUFs)*. A Unique Object is a physical system that, upon measurement by an external apparatus, exhibits a small, fixed set of inimitable analog properties that are not similar to any other objects. It shall be impossible to intentionally fabricate a second object with the same properties, even if the properties and exact structure of the original object are known. Such properties can be referred to as the "fingerprint" of a unique object for obvious reasons. We discuss several media that exhibit such unique disorder, including paper, fibers, magnetic disks, radiowave scatterers, and optical tokens.

PUFs are the second important class of disordered systems that can be employed for reliable identification, authentication, key storage, and other security tasks. The term and acronym "PUF" for denomination of this class first appeared in [1]. In a nutshell, a PUF is a disordered physical system S that, when interrogated by a *challenge (or input, stimulus)* denoted by C_i, generates a unique device *response (or output)* denoted by R_{C_i}. This response shall depend on the applied challenge and on the specific disorder and device structure of the PUF. The unclonability requirement in the PUF definition is that it should be intractable for an adversary with physical access to create a physical or software clone of a PUF.

Both the challenge-response pairs of PUFs and the fingerprints of Unique Objects have the purpose of uniquely identifying any device with high probability. In order to realize this in practice, we need stable repeated measurements, and must be able to cope with noise and varying operational conditions. In such scenarios, error correcting codes may be used to ensure the desired operability and robustness of the system [2–7]. Other options are averaging or calibrating the device's operational conditions [8, 9].

Two important metrics that are typically applied to categorize the uniqueness and robustness of PUF responses and UNO fingerprints are *inter-device* and *intra-device* distances. Inter-device distance is often quantified as the average Hamming distance between the responses to the same challenge obtained from two different PUFs/UNOs, or the average distance between the fingerprints of two unique objects measured in the same conditions. Intra-device distance is the average Hamming distance between the responses to the same challenge applied at different times and environmental conditions to the same PUF/UNO, or the average distance between the repeatedly measured fingerprint(s) of a unique object. Ideal PUFs and UNOs should lead to large inter-device and small intra-device distances. Another key requirement for PUFs and unique objects is the entropy of the resulting responses or fingerprints. The entropy quantifies the number of independent IDs that can be generated by the same device architecture.

Despite the similarities between UNOs and PUFs, there are several important differences between them that distinguish these two security primitives (and their subclasses) from each other. This chapter provides a conceptual categorization and summary of the field of physical disorder based cryptography and security, also termed *physical cryptography* in [10]. Whenever applicable, the concepts are interleaved with examples from the contemporary literature and implementation details. A number of surveys on the PUF subject are already existent, for example, [11] and a recent book with several chapters dedicated to PUFs [12]. We will cite these review articles whenever applicable, and emphasize that the concepts in this chapter are complementary to the contemporary literature in this area.

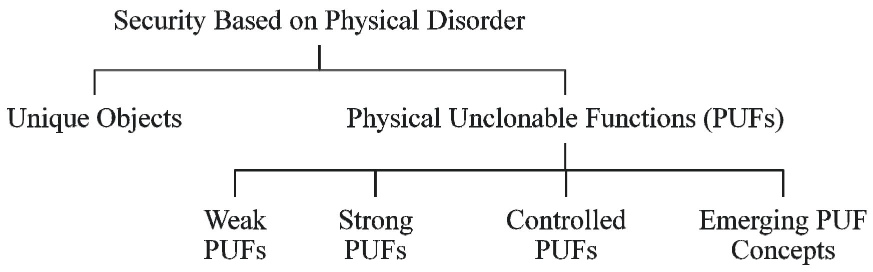

Security Based on Physical Disorder
Unique Objects
Physical Unclonable Functions (PUFs)
Weak PUFs
Strong PUFs
Controlled PUFs
Emerging PUF Concepts

Fig. 1. Organization of the Chapter.

Organization of this chapter. Figure 1 gives an overview of the classes of physical disorder based security tokens discussed in this chapter. Each of the discussed subjects are shown as a branch in the chart. The next section reviews UNOs including paper-based (fiber-based) fingerprints, magnetic signatures, and RF-based Certificates of Authenticity. Section 3 discusses the weak PUF class including Physically-Obfuscated Keys, SRAM-PUFs, and butterfly PUFs. Strong PUFs are the subject of Section 4. Examples of PUF structures that can provide building blocks for Strong PUFs include optical PUFs, arbiter PUFs, XOR arbiter PUFs, and analog cellular arrays. Emerging PUF designs and research challenges are presented in Section 6. We conclude the chapter in Section 8.

2 Unique Objects

Extracting an objects's fingerprint based on its random physical disorder has been exploited for more than three decades. In absence of an established common term, we call this class of structures "Unique Objects (UNOs)".

A Unique Object is a physical entity that exhibits a small, fixed set of unique analog properties (the "fingerprint") upon being measured by an external equipment. It should be possible to measure the fingerprint quickly and preferably by an inexpensive device. The "fingerprint" should be specific to the object such that it is practically infeasible to find or build another instance of the object with the same specs, even if the object's fingerprint and its detailed structure are known (see also [13]).

More precisely, a Unique Object and its fingerprint should meet the following properties:

1. *Disorder.* The fingerprint should be based on the unique disorder of the physical object.
2. *Operability.* The fingerprint should be adequately stable over time, and must be robust to aging, environmental conditions, and repeated measurements. It must be possible to fabricate other instances of the measurement equipment with similar characterization capability. The measurement and characterization cost and time should be practically and economically viable.
3. *Unclonability.* It should be prohibitively expensive or impractical for any entity (including the manufacturer) to produce another object that presents the same unique fingerprint characteristics when queried by the measurement device.

Figure 2 demonstrates the scenario. It is assumed that each Unique Object in the figure has an unclonable fingerprint that is specific to it. Also, it is assumed that both measurement equipment are able to characterize the object's fingerprint at the desired level of resolution and accuracy. In other words, the UNO shall be the unique and unclonable part of the system, while the measurement device can be mass-produced with the same functionality.

2.1 History and Examples of Unique Objects

Using biometrics for fingerprinting dates back to the 19th century. Although human hand fingerprints and other biometric entities are closely related to Unique Objects, a discussion of biometrics fingerprinting is outside the scope of this chapter. We refer the interested readers to comprehensive books on the subject [14, 15].

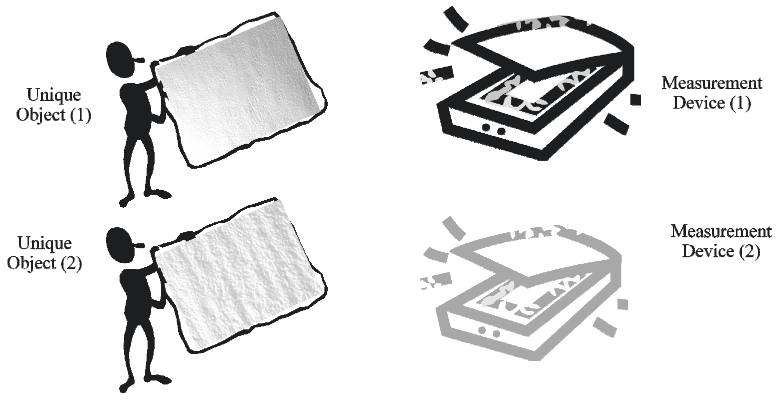


Fig. 2. Two Unique Objects (based on paper structures in this example), and two fingerprint measurement devices. The cloning of a Unique Object should be prohibitively costly, while it should be possible to mass-manufacture large numbers of measurement devices that can characterize the fingerprints at the desired level of accuracy.

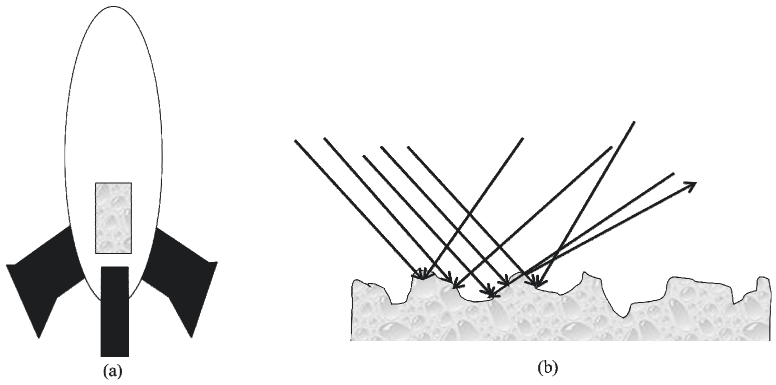


Fig. 3. (a) A thin, random layer of light scattering particles sprayed on the missiles. (b) Illuminating the surface from different angles would generate different inference patterns.

Sprayed random surfaces. Perhaps the earliest reported usage of Unique Objects for security was proposed by Bauder during the cold war for use in nuclear weapons treaties [16, 17]. To tag the nuclear missiles unforgeably, a thin, random layer of light scattering particles was sprayed onto the missiles. The layer was illuminated from various angles, and images of the resulting interference patterns were recorded by an inspector. On later inspections, the interference patterns could be measured anew, and compared to the record of the inspector. An example is shown in Figure 3.

The scheme was assumed secure even if an adversary would know the (relatively few) illumination angles and the resulting patterns used by the inspector, and even if he had access to the unique layer for a long time period and could investigate its structure. Even under these circumstances, it was presumed infeasible to produce a second layer which generated the same speckle pattern. Of course, if an adversary knows all illumination angles and the resulting patterns used by the inspector, this system cannot be used for remote authentication, since an adversary can merely replay back the digitized responses/images upon receiving a challenge. Furthermore, the scheme can only be used by an inspector who carries trusted measurement equipment and uses it on the Unique Object directly, which was presumably the usage model of the system during the cold war.

Fiber-based random surfaces. Other early implementations of Unique Objects were based on randomly distributed fibers in solid-state objects, for example, paper fibers in banknotes [18], or metallic fibers in a thin substrate on bank cards measured by a magnetic reader [19]. A seminal reference that discusses Unique Objects from a more fundamental cryptographic perspective is [20], which was later extended by [21]. [20] is, to our knowledge, the first academic source that suggests the combined use of Unique Objects and digital signatures to create offline verifiable labels.

Several seeds laid in this early work were followed up in later research. Firstly, surface irregularities in paper or other materials were further investigated by [22–28]. The authors of [23, 24] create unforgeable postal stamps and document authenticators from paper irregularities and digital signatures; [22] shows that the paper surface fingerprints are robust to soaking and a number of other transformations;

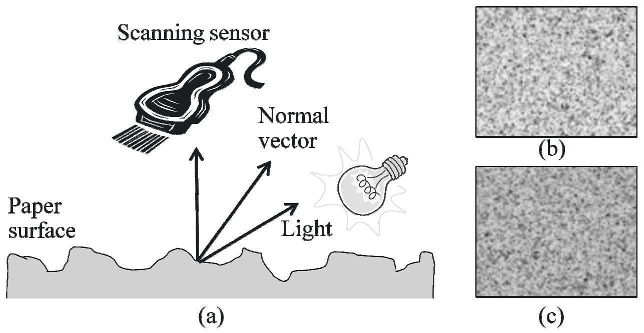


Fig. 4. (a) Scanner produces different images of the paper surface based on the page orientation. The light seen at the sensor depends on the angle between the surface normal and the light source; (b) A region of the document can be scanned from top-to-bottom; and (c) The same document region can be scanned from left-to-right. The 3D texture can be estimated by combining (b) and (c) (Figure inspired by studies in [28]).

[26] investigates the use of surface-based labels in developing countries and rural regions; [27] deals with database and error correcting algorithms for surface-based identification; and [28] offers a detailed treatment centering around inexpensive measurement methods for paper surfaces by commodity scanners as demonstrated in Figure 4. The complex reflective behavior of surfaces has even led to commercially available security systems [29] [30].

Secondly, the use of randomly disordered fibers contained in a fixing matrix was described in [31–34]. [32, 33] use light conducting optical fibers, and measure the individual light transfer via these fibers into various spatial segments of the matrix. Each instance is created as a collection of fibers randomly positioned in an object by a transparent gluing material that permanently fixes the fibers' positions. Readout of the random structure of a fiber-based note is performed using the following fact: if one end of a fiber is illuminated, the other end will also be lit as shown in Figure 5.

[31] employs randomly distributed metal particles, and measures their scattering behavior by a near-field read-out. [34] pours ultraviolet fibers into a paper mixture and measures the optical response.

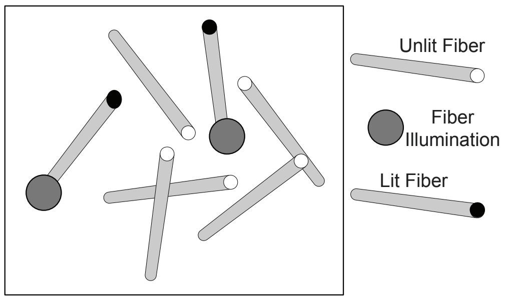


Fig. 5. Examples of randomly placed, fixed-length fibers. The square demonstrates the embedding substrate. Three fibers lit by spot illumination light as described in [33].

Unique Objects and digital rights management. An observation that further propelled the field was that common data carriers such as (again) paper, but also ICs, CDs, and DVDs can have unique features. Sometimes their unique properties arise just in the very process of writing or imprinting data onto them. One early reference in this context is [35]. There, the unique irregularities in CD-imprinting (such as individual height, shape and length of the bumps) are used to secure the CD's digital content that is stored in exactly these bumps. Conceptually the same suggestion is made in [36] and [37], yet at a much greater level of scientific detail. For more information on optical media (CD) fingerprints, we refer interested readers to [12]. The irregularities in letters printed on paper have been suggested to secure paper documents in [38]. Finally, several methods for uniquely identifying and authenticating chips will be described in the remainder of this chapter.

Other implementations of Unique Objects. Other studies proposed novel classes of unique structures, and can be best categorized according to the employed read-out technique. A number of Unique Objects with radio wave read-out in the (relative) far field were suggested in the commercial sector [39–44]. For most of them, doubts have been raised with respect to their unclonability [13]. Another radio wave approach measures the unique RF signals created by higher harmonic oscillations [45]. Unique Objects with magnetic read-out have been investigated in [46]. Alternative optical concepts that dig deeper into physics and utilize more complex effects and structures have been suggested in [47]. They include photonic crystals and resonant energy transfer between optically-active particles. Surprisingly, there is little work on Unique Objects with electrical read-out, even though this would promise particularly simple and inexpensive measurement. One recent source is [10], where the unique current-voltage characteristics of imperfect diodes are exploited. Finally, even DNA-based approaches have been suggested [48], and made it to the marketplace some time ago [49].

Finally, the question of error correction of the measured unique signals is treated en passant in most of the above publications, including [27, 28, 32, 34]. References solely dedicated to error correction include [50–52].

2.2 Protocols and Applications of Unique Objects

The natural application of Unique Objects is to label items of value (such as commercial products, bank cards, banknotes, passports, access cards, etc.) in an unforgeable manner. Proving authenticity is particularly useful as losses due to counterfeiting of digital goods and physical objects amount to worldwide losses in a three-digit billion dollar range [53]. Two basic approaches can be applied.

(i) In one classic and straightforward approach, the Unique Object is physically attached to the item it protects, or consists of a measurable unique characteristic of the protected item itself. The Unique Object's fingerprint is stored in a central database. When authentication is needed, the object's fingerprint is measured and compared to the stored value in the database. The requirements for this protocol include existence of a central database and an authenticated online connection to the database.

(ii) An alternative approach has been pioneered, to our knowledge, in [20], and has been termed Certificate of Authenticity (COA) in [31]. Again, the Unique Object is physically attached to the protected item (or is a measurable unique characteristic of the protected item itself). In addition to the Unique Object, complementary information is stored directly on the item, for example via a printed barcode. The information includes a numerical encoding of the fingerprint, error-correcting codes [52], item-related information I (such as the origin of the item), and most importantly, a digital signature of the fingerprint and I. In order to verify the validity of the label/the item, a verification device does the following: It reads the complementary information from the item, and verifies the validity of the digital signature by use of a public verification key stored in the device. Secondly, it measures the fingerprint of the label/the item by a trusted measurement apparatus, and verifies if it matches the fingerprint given and signed in the complementary information.

The advantage of the approach (ii) is that it does not need a connection to a database and that it can work offline. Neither the label nor the testing apparatus needs to contain secret information of any sort. This leads to the strong asset that neither tampering with a label nor with any of the widespread testing apparatuses can lead to secret key extraction and a global failure of the system through one extracted key. The measurement apparatus has to be trusted to work correctly.

Variants and combinations of the two basic protocols above have been proposed in Unique Objects literature, e.g., [35–37].

2.3 Security Features

No secret information in hardware. The most striking feature of Unique Objects is that they contain no piece of information that must remain secret, and which would have to be protected by costly and laborious means. Their security rests not on the assumption that some key or some other information about their structure remains secret; rather, it is built on the hypothesis that it is infeasible to build a clone of the object even if all its internal details are known. It is based directly on the limitations of current nano-scale fabrication methods.

Furthermore, a COA can even be verified for validity without possession of any secret keys; any verifying party merely must hold a public key to check the digital signature contained in the COA. This allows the widespread distribution of labels and testing apparatuses, without risking a global security break through secret key compromise in either the labels or apparatuses, which is significant.

The only secret key that is required can be stored at the authority that creates the signatures, where it can usually be protected much better. The authenticated communication required in classic protocol (i) above can be established by typical cryptographic methods. At the same time, parties using the system must rely on the integrity of the measurement apparatus. This implies that remote authentication to a central authority by an untrusted terminal is not possible, and therefore limits applicability of Unique Objects.

Structural sensitivity as a security benchmark. One critical measure for the security of Unique Objects is their structural sensitivity: How much are the output signal and the unique measured fingerprints of the object affected if we change its inner structure slightly, by a factor of δ, say? This parameter determines the level of exactness that an adversary has to reproduce the Unique Object in order to remain undetected. It can be employed as a benchmark to rank competing candidates of Unique Objects.

Attacks on Unique Objects. The main attack on Unique Objects is refabrication or cloning. It is not necessary to rebuild the original with perfect precision; merely, a second structure needs to be fabricated that generates the same measured fingerprint as the original from the view of the measurement apparatus. This structure could in principle have a totally different size, lengthscale, or appearance; it might even be a smart, reactive system that artificially synthesizes the correct response. Note that purely numerical modeling attacks such as the ones executed on PUFs [54] are pointless and not applicable to Unique Objects. Such attacks can help a fraudster to numerically predict the (numerical) response of a PUF to a randomly chosen challenge. But in case of UNOs, the attacker is assumed to know these responses anyway; his task lies in fabricating a clone that produces the same analog response upon measurement with an external apparatus that he/she cannot influence. This is foremost a physical manufacturing challenge, not a question of modeling.

Quantum systems vs. Unique Objects. Quantum systems, such as polarized photons, were among the first systems whose inherent physical features have been suggested for security systems [55] [56]. It is long known that the laws of quantum physics forbid the cloning of a quantum system with an unknown state, for example of a photon with an unknown polarization. Could a polarized photon hence be interpreted as a specific object according to our definition, with its unique property being the polarization angle? This is not the case: One condition of the Unique Object definition is that the adversary knows the unique properties of a Unique Object. But once the polarization of the photon is known, many photons with the same polarization state can be generated. Unique Objects thus relate on a different type of unclonability than quantum systems.

3 Weak Physical Unclonable Functions (Weak PUFs)

One class of Physical Unclonable Functions based on inherent device variations are Weak PUFs. They exploit the disordered, unique, internal structure of the underlying fabric as a non-volatile memory for storing the secret keys. In an ideal case, the volatile keys generated by Weak PUFs upon power-up cannot be determined by external and invasive attacks due to construction or tamper-proof properties of the pertinent structure. Weak PUFs are also known under the name of Physically Obfuscated Keys (POKs) [2].

The term *Weak PUF* was coined in [57] to refer to PUFs with a limited number of challenge-response pairs (CRPs) in contrast to Strong PUFs that contain many CRPs. The following specification has it in greater detail.

1. *Challenge-Response Pairs.* A Weak PUF can be interrogated by one (or a very small number of) fixed challenge(s) C_i, upon which it generates response(s) R_{C_i} that depends on its internal physical disorder.
2. *Key Derivation.* The response(s) R_{C_i} from a Weak PUF is (are) exploited by the device for deriving a standard digital key that can be used for security applications.
3. *Practicality and operability.* The generated response R_{C_i} should be sufficiently stable and robust to environmental conditions and multiple readings.

Weak PUFs vs. UNOs. It is important and necessary to differentiate Weak PUFs from Unique Objects: Applications of Unique Objects require an adversarial model where Eve has time to inspect all features of the Unique Object, and will often know its internal structure and unique fingerprint. Furthermore, these unique properties are measured by an external apparatus. The exact opposite holds for Weak PUFs: Their responses are measured internally, and the derived key is kept secret in the embedding hardware.

3.1 History and Implementation Examples

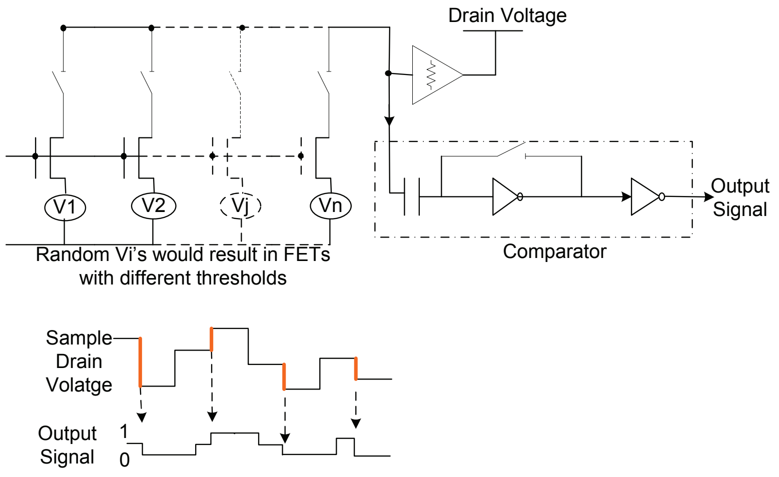


Fig. 6. Array of ICID transistors producing a sequential random voltage proposed in [58].

ICID PUFs. ICID is the first proposed and designed circuit structure for generating a Weak PUF (or *random chip ID*) based on process variations [58]. They devised an array of addressable MOSFETs (shown in Figure 6), with common gate and source and sequentially selected drains driving a resistive load. Because of device threshold voltage mismatches (resulting from process variation) the drain currents are randomly different. Therefore, at each die, a unique sequence of random voltages would be generated at the load. ICID exploits these unique sequences of random but repeatable voltages to construct unique identification. In $0.35\mu m$ technology, the authors reported about 10% false positive and false negative results for repeating random bits on their test circuits. Identification capability can be improved by increasing the bit length.

Physically Obfuscated Keys (POKs). Under the name of a Physically Obfuscated Key (POK), Gassend proposed a type of Weak PUF that was built from the first integrated Strong PUF (see Figure 8 in Section 3.2 for the architecture, and see Section 4 for Strong PUF) [2]. The POK/Weak PUF would only utilize one (or a small subset) of all possible challenges for a Strong PUF. This allows using them exactly as a digital key that is more resistant to physical attack, because it extracts its information from a complex physical system.

SRAM-based PUFs. A commonly used candidate structure for a Weak PUF exploits the positive feedback loop in an SRAM or an SRAM-like structure. If a write operation is used, the cross-coupled device starts transitioning to the inserted value, and the transition is sped up by the positive feedback loop in the structure. When no write operation is in place and the system is not in any of the states (initial power up), the inherent small transistor threshold mismatches and thermal and shot noise trigger the positive feedback loop so the state would be in one of its two possible stable points (0 or 1). The effects of common mode process variations including lithography, and common mode noise (e.g., substrate temperature and supply fluctuations) is similar on the differential device structure and does not strongly impact the transition.

The idea of fingerprinting of semiconductor integrated circuits using SRAM was originally suggested in a 2002 patent, but no experimental implementation data were included [59]. The work in [60] constructed a custom-built array of SRAM-like cells that generated random values based on threshold mismatches in $0.13\mu m$ technology (Figure 7). Their experiments have shown a close to uniform distribution of the bits (close to Normal distribution of Hamming distances) and more than 95% bit stability. The work in [61] showed that initialization of SRAM can produce a physical fingerprint for

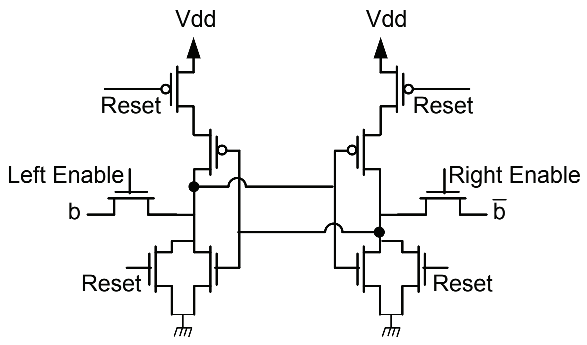


Fig. 7. The positive feedback loop created by the cross-coupled NOR (NAND) gates is used for storing a 0 or a 1 in SRAM-like memory structures.

each chip. They have also shown that the fingerprints can pass the standard NIST randomness tests for runs [61]. The authors in [57] also exploit the initial state of the SRAMs in an FPGA to extract IDs based on differential device mismatches. They coined the term *intrinsic PUF* to refer to structures that do not need additional circuitry for embedding the PUF.

Since not all FPGAs have SRAMs built in them, the work in [5] proposed *butterfly PUFs* based on reconfiguring the FPGA cells to construct two back-to-back NAND gates (in positive feedback mode) similar to the SRAM structure. Note that the Butterfly PUF cannot be considered intrinsic, since it should be custom configured the same way as any other logic circuitry can be made on a reconfigurable fabric.

Coating PUFs. Another construction of a Weak PUF is a *coating PUF* that provides a practical implementation of read-proof hardware. A read-proof hardware device has the property that once constructed, no outside entity can read (extract) information on the data stored in the device. The authors in [62] introduced coating PUFs as form of a protective coating that can be sprayed on the IC and cover its surface. The coating is composed of a matrix material doped with random dielectric particles (i.e., different kinds of particles of random shape, size and location with a relative dielectric constant differing from the coating matrix's dielectric constant). The top metal layer of the IC contains an array of sensors that are used to measure the local capacitance values of the coating.

One central property of a Coating PUF is their purported tamper sensitivity: It is assumed that any tampering with the coating (such as invasive penetration, or removal from the covered IC) strongly and irrecoverably changes its properties. [62] has positively evaluated the resilience of coating PUFs against some optical and invasive attacks.

Resistive PUFs. Another instance of silicon-based PUFs are based on power distribution and resistance variation of chips that have appeared in recent literature [63, 64].

3.2 Protocols, Applications, and Security

Secret key generation and storage. Weak PUFs provide a method for secret key generation and storage based on random disordered physical medium fluctuations. Therefore, any security protocol that leverages the storage of a secret key can utilize a Weak PUF in its flow. To our knowledge, the earliest security protocols and IP protection applications based on Weak PUFs/POKs were presented in [2, 65]. Other protocols and applications including metering and RFID protection based on Weak PUFs were presented in [62, 66, 57, 5, 67].

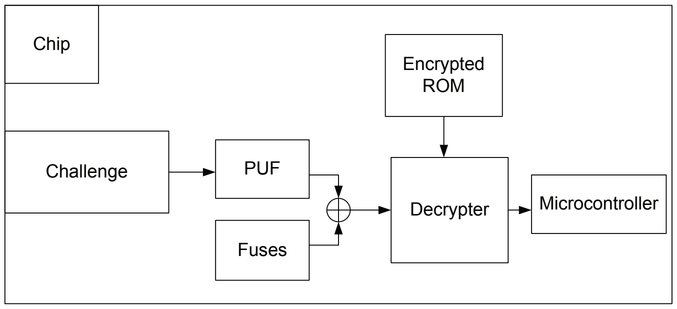


Fig. 8. A POK built by using a Strong PUF proposed in [2].

IP protection application. Weak PUFs were proposed for protecting hardware IP and ICs against piracy. A proposed system for protecting programmable hardware IP against piracy is shown in Figure 8 taken from [2]. Assume that the design is a microcontroller with a compression algorithm stored in ROM. A Strong PUF is hardwired with other functions on the chip to generate a k-bit key K that is the same for all chips to mitigate the cost. The challenges to the PUF are also hardwired to be fixed. That PUF response is combined with the contents of burned on-chip fuses through an exclusive-or operation to produce K. A decoder uses K to decrypt the ROM content. By selecting the fuse bits one can generate the same decrypting key K on all chips. The response never leaves the chip during the decryption operation. Even if the state of all the fuses are discovered, the key would remain secret.

Secure processor. Suh [65] describes how a Weak PUF can be embedded in a secure processor which then can be used for applications such as certified execution and software licensing. In one design, the weak PUF is used to generate a seed for a public/private key pair. The seed and private key are never exposed and the public key is published and certified by a certification authority. In another, the seed is used as a symmetric key to encrypt a secondary symmetric key that is known to the user of the processor. Again, the seed remains unknown, and is only used to encrypt a given secondary key and decrypt the secondary key for internal use in secure execution.

Active IC metering. Another usage of Weak PUFs was for active IC metering that protects the hardware against foundry piracy (overbuilding) [66]. Here, the functional specification of the design in the finite state machine (FSM) domain was modified. The alteration was such that an exponential number of states were added to the design with a low overhead. Hiding a state in the large state-space of the FSM was later shown to be an instance of a provably obfuscatable general output multi-point function. It was shown that the transitions from the hidden state cannot be found by having access to the layout and even access to the register's contents that store the state. Upon fabrication at the foundry, based on the Weak PUF's response, the design would be in one of the hidden obfuscatable states that is called a locked state. This locked state can be read out by everybody, but the passkeys to the functional (unlocked) state can only be provided by the original designer who has access to the modified FSM.

Security analysis. Weak PUFs are commonly attributed three advantages:

(1) They are harder to read-out than standard digital keys that are stored permanently in non-volatile memory (NVM) since keys only exist when the chip is powered.
(2) They can possess some natural tamper sensitivity, meaning that any tampering with the device, or even with the hardware system that embeds the PUF, alters the physical features of the device and the key derived from them.
(3) They save on costs, since they avoid the process steps necessary to include NVM in hardware systems.

Some of these assets must be analyzed further. Let us start with advantage (1): Weak PUFs clearly avoid the *long-term* presence of *digital* keys in NVM. But the security of a Weak PUF based hardware still depends on the secrecy of a single digital key derived from the Weak PUF, which is present for at least a short period after its derivation from the PUF's responses. This creates a single digital point of failure for the system. Weak PUFs furthermore cannot alleviate the permanent presence of secret information in the hardware in general: If an adversary knows the disorder or fabrication mismatches that determine the responses of the Weak PUF, he may simulate and derive these responses.

Further, Weak PUF based hardware may suffer from similar weak spots as other systems built on standard binary keys. Side channel or emanation analysis may be possible; and since the device will apply some standard cryptoprimitives to K, its security will thus depend on the same unproven computational assumptions as any classical system built on digital keys.

Regarding the above asset (3), it must be stressed that error correcting information is vital for Weak PUFs; any single bit flips make the system not applicable any more. This necessitates the use of accompanying error correcting information, which must be stored in NVM of some form. To the asset of Weak PUFs, this storage can be external and/or public; further, it need not be implemented in the hardware that contains the Weak PUF.

4 Strong Physical Unclonable Functions (Strong PUFs)

Immediately after the introduction of Weak PUFs or POKs, a second class of PUFs was put forward [68, 69, 1, 70]. They have later often been referred to as Strong PUFs, for example in [57, 71, 72].

In a nutshell, a Strong PUF is a disordered physical system with a very complex input-output behavior that depends on its disorder. The system must allow very many possible inputs or challenges, and must react with outputs or responses that are a function of the applied challenge and of the specific disorder present in the system. The input/output behavior should be so complex that it cannot be imitated numerically or by any other device.

More specifically, a Strong PUF is a disordered physical system S with the following features:

1. *Challenge-Response Pairs.* The Strong PUF can be interrogated by challenges C_i, upon which it generates a response R_{C_i} that depends on its internal physical disorder and the incident challenge. The number of CRPs must be very large; often (but not always) it is exponential with respect to some system parameter, for example with respect to the number of components used for building the PUF.
2. *Practicality and operability.* The CRPs should be sufficiently stable and robust to environmental conditions and multiple readings.
3. *Access mode.* Any entity that has access to the Strong PUF can apply multiple challenges to it and can read out the corresponding responses. There is no protected, controlled or restricted access to the PUF's challenges and responses.
4. *Security.* Without physically possessing a Strong PUF, neither an adversary nor the PUF's manufacturer can correctly predict the response to a randomly chosen challenge with a high probability. This shall hold even if both parties had access to the Strong PUF at an earlier time for a significant period, and could make any

reasonable physical measurements on the PUF, including (but not limited to) determination of many CRPs.

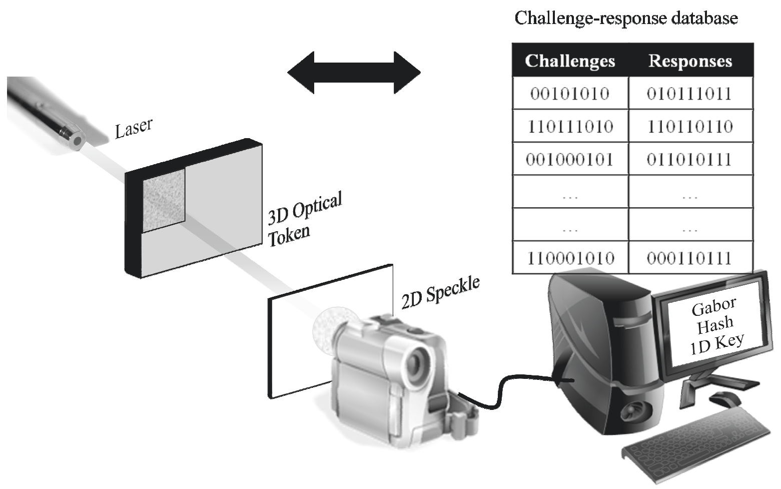
The definition above is more qualitative than quantitative in order to remain intuitive; a more formal and thorough definition can be found in [72].

Unique vs. Weak vs. Strong. While a Unique Object must always possess an external and a Weak PUF always an internal measurement equipment, this is left open for Strong PUFs; both variants are possible and have been realized in practice (see [68, 69] for an optical PUF with an external and [1, 73] for an electrical PUF with an internal measurement apparatus). Unique Objects require a trusted measurement apparatus whereas Strong PUFs once "bootstrapped" (cf. Section 4.2) can be remotely authenticated with an untrusted measurement apparatus. Another difference between Strong PUFs and Unique Objects lies in the exact adversarial model and the relevant security properties: While the adversary's aim in the case of Unique Objects lies in physically fabricating a clone device with the same properties, his goal in the case of Strong PUFs is to learn how to predict the input/output behavior of the Strong PUF. The latter is a mixture of numerical assumptions and physical hypotheses. This fact does not exclude that the same structure can be used as a Unique Object and as a Strong PUF under different read-out schemes, for example; but not every Unique Object is a Strong PUF and vice versa.

Weak PUFs possess only a small number of fixed challenges, whereas Strong PUFs have a very large number of challenges. In Weak PUFs, the responses remain secret and internal. To the contrary, Strong PUFs allow free querying of their responses.

4.1 History and Examples of Strong PUFs

Optical PUF. The first implementation of a Strong PUF has been suggested in [68] [69], albeit under the different name of a physical one-way function. It consists of a transparent plastic token which contains a large number of randomly distributed glass spheres as shown in Figure 9. We call this implementation an *optical PUF*. An individual, unclonable token is illuminated under different angles



Challenge-response database

Challenges	Responses
00101010	010111011
110111010	110110110
001000101	011010111
...	...
...	...
110001010	000110111

Fig. 9. A 3D inhomogeneous transparent plastic token being optically challenged (illuminated under different angles and points of incidents) and produces an output in form of an interference pattern. The output is hashed to produce a 2D image, which is in turn filtered by a multiscale Gabor transform to produce a 1D key as proposed in [69].

and points of incidence (which are regarded as the challenges of the system), and produces an interference pattern, which is considered the response of the system. We draw the reader's attention to the similarity in Figures 3 and 9. The main difference is a usage one: optical PUFs are assumed to have a large number of challenges, and a secret set of challenge-response pairs is stored in a central database. Thus, optical PUFs can be remotely authenticated.

This construction is presumably secure (no attacks are known to this date), but the measurement apparatus is external and relatively large, potentially leading to practicality issues and stability problems when the token is measured by different apparatuses at different locations.

Arbiter PUF. Almost simultaneously to optical PUFs, the first integrated electrical Strong PUFs including "Arbiter PUFs" were put forward in [1] [73]. [1] is also the first publication that uses the term PUF as a common abbreviation for the expressions Physical Random Function and Physical Unclonable Function. Unlike optical

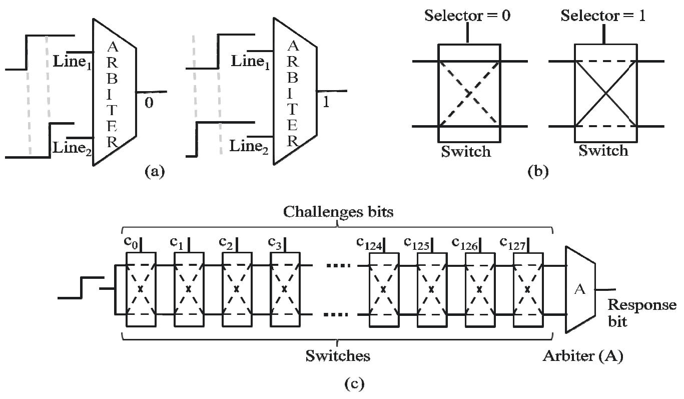


Fig. 10. (a) Demonstration of an arbiter's operation: the relative time of signal arrival at Line$_1$ and Line$_2$ would determine the value of the output bit; (b) Demonstration of a selector's operation: the selector bit would decide if the top and bottom lines continue in the same order, or they switch places; (c) An arbiter PUF with 128 challenge bits $c_0, \ldots, c_{127}$ applied as the selectors to the switches. The switch selectors dynamically configure two parallel paths with random delay differences that would form the response generated by the arbiter [74]).

PUFs, silicon PUFs do $\underline{\text{not}}$ require external measurement equipment. They are based on the runtime delay variations in electrical circuits.

In one implementation, an electrical signal is split into two parallel signals, which race against each other through a sequence of k electrical components, for example, k multiplexers. This architecture is shown in Figure 10. As shown in the figure, the challenges are applied to the selectors of the multiplexers. The exact signal paths are determined by these challenge bits $b_1, \ldots, b_k$ applied at the multiplexers. At the end of the k components, an arbiter element decides which of the two signals arrived first and correspondingly outputs a zero or a one, which is regarded as the system's response.

It was clear from the beginning that these first electrical candidates were prone to modeling attacks as mentioned in [1]. Attacks using machine learning algorithms have been carried out, see Section 4.2. In these attacks, the adversary collects many challenge-response pairs (CRPs), and uses them to derive the runtime delays occurring in the subcomponents of the electrical circuit. Once they are known, simple simulation and prediction of the PUF becomes possible, breaking its security. One reason why these attacks worked so well lies in the fact that plain Arbiter PUFs have relatively simple linear models, in which the delay of each of the two signals can be approximated as the linear sum of the delays in the subcomponents. This makes standard machine learning algorithms applicable to the problem.

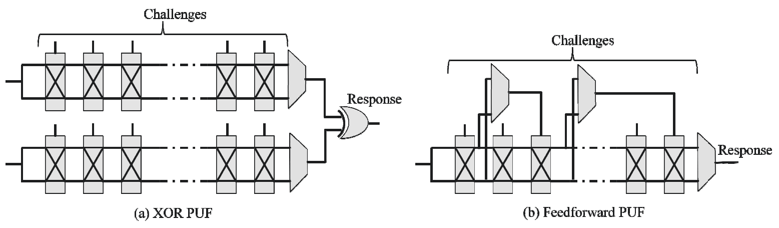


Fig. 11. (a) An arbiter PUF wtih added XORing of two arbiter outputs; (b) Feedforward PUF.

Variants of the Arbiter PUF. The above issues naturally led to the introduction of non-linear electrical PUFs, for example, XOR arbiter PUFs, Lightweight Secure PUFs and Feedforward Arbiter PUFs [11, 75, 76, 74]. In an XOR arbiter PUF, multiple arbiter outputs are XOR'ed to form a response. In Figure 11(a), an example is shown where two arbiter outputs are XOR'ed. In the Feedforward Arbiter PUF, the output of intermediate multiplexer(s) on the signal paths are input to so called Feedforward arbiter(s). The Feedforward arbiter output is then fed to the input of another multiplexer forward on the signal path. In Figure 11(b), an example of a Feedforward arbiter structure is shown. All of the aforementioned structures employ the basic Arbiter PUF architecture, but refine its architecture by introducing additional, non-linearities. These structures showed a significantly higher resilience against machine learning attacks, but still could be attacked up to a certain level of size and complexity [77, 54].

Arbiter PUFs and their variants have been shown to have small and stable integrated electrical implementations and have been commercialized [78].

Legacy PUFs. [80] has proposed using the ICs' timing path signatures that are unique for each state-of-the-art CMOS chip (because

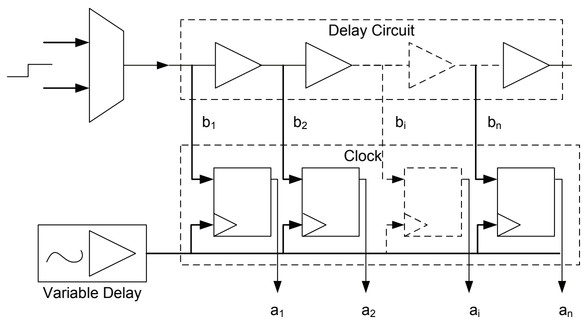


Fig. 12. The glitch PUF architecture samples the glitches on the path and the arrival of a glitch compared to a clock signal generates the response bits in the FFs [79].

of process variations) as a PUF. The work in [81, 82] has shown that all ICs that are fabricated in new CMOS process nodes that contain nontrivial process variations have a unique signature that can be extracted using noninvasive methods by the structural side channel tests such as IDDT, IDDQ, or delay tests. They have shown a unified gate-level characterization of the signatures for all side-channels that could be used as a compact representation. It was shown that statistical signal processing methods can be adopted for ensuring rapid and robust characterization [83–87]. The interesting aspect of this line of work is that the signatures are intrinsic to all legacy ICs, and there is no need for insertion of additional circuits or structures by the manufacturer or other parties who are interested in verifying the chip's authenticity by its specific signature. Therefore, it can be readily used for digital rights management of integrated circuits in the supply chain and for anti-counterfeiting protection.

Analog PUF family. New, recent suggestions for Strong PUFs have tried to exploit the analog characteristics of electrical signals, such as in analog cellular arrays [88]. The system suggested in [88] imitates optical wave propagation in an electrical cellular non-linear network, transferring the known complexity of optical PUFs into electrical circuits. Another non-linear electrical suggestion is [79] that is based on the nonlinear propagation of glitches on a logic path. Figure 12 demonstrates the architecture of the glitch PUF system, where the glitches based on the delay difference between the signal path and the clock signal are stored in the response FFs.

Finally, integrated optical PUFs have been proposed [89], but their security seems suspect if merely linear scattering media are used (see appendix of [90]).

4.2 Protocols, Applications, and Security

Protocols and applications. The archetypical application of Strong PUFs is the identification and authentication of hardware systems (or other security tokens such as credit cards) [69] [68] [1]. The corresponding protocols are usually run between a central authority (CA) and a hardware/token carrying a Strong 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 using a trusted external measurement apparatus in the case, for example, of an optical PUF. This step is usually referred to as bootstrapping. Whenever the hardware, possibly at a remote location, 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 in the clear. The hardware applies these challenges to S, and sends the obtained responses to the CA, also in the clear. If these responses closely match the pre-recorded responses in the CRP-list, the CA believes the identity of the hardware. Note that each CRP can only be used once, whence the CRP-list shrinks over time, and needs to be large. As noted above, an exact match is not required, and a certain level of noise in the responses can be tolerated.

Another application that has been mentioned is key exchange or key establishment based on Strong PUFs [69]; a formal protocol has been given in [89]. However, it has been shown in [91] that such key exchange protocols can suffer from problems regarding their forward secrecy and their repeated use for session key exchange. [91] also proposed a new type of PUF that can fix this issue, called erasable PUFs. We note that this new type of erasable PUFs is different than the earlier FPGA PUFs that could be configured and erased for each authentication session [77, 92].

The above protocols give Strong PUFs broad cryptographic applicability. They can be employed for any application which requires the above cryptographic tasks, often without storing explicit digital keys in the hardware containing the PUF.

Security features and Attacks. Attacks on Strong PUFs will either try to build a physical clone, i.e., a second physical system that behaves indistinguishably from the original PUF, or a digital clone, i.e., a computer algorithm that imitates the PUF's challenge-response behavior.

It has been rightfully stressed in early publications on Strong PUFs [68, 69] that they can avoid the classical, well-known number-theoretic assumptions in cryptographic protocols. But is the security of Strong PUFs entirely free of computational assumptions, and can it merely be built on their internal entropy and randomness? It is known that the maximal amount of randomness or entropy in a physical system is polynomially bounded in the size of the system [93, 71, 72]. This implies that the overall number of possible challenges of many PUFs is larger than their entropy. In particular, this observation necessarily holds for any PUFs with an exponential number of challenges.

An adversary therefore often merely needs to gather a small subset of all CRPs of a Strong PUF to obtain (at least indirect) knowledge about all CRP-relevant information/entropy contained in the PUF. Once he has gathered such a subset, it is merely a computational assumption that he cannot derive an internal PUF model from it which allows PUF prediction. For example, he could set up a system of (in-)equations from the CRP subset, whose variables describe the inner PUF structure. If he can solve this system of (in-)equations efficiently, he can break the PUF. The hypothesis that he will not be able to do so is just another type of unproven computational assumption.

This perhaps surprising observation is not just a theoretical concern. The modeling attacks presented in [2, 75, 94–96, 71, 77, 54] are practical, and prove the basic feasibility and effectiveness of such attacks. They also exhibit that such attacks reach their limits when the involved computations become too complex; for example, the authors of [54] could not attack XOR arbiter PUFs with $k > 6$ XORs because the complexity grew exponentially in k.

In other words, the security of many Strong PUFs is dependent on the underlying computational assumptions. In favor of Strong PUFs, it must be said that these assumptions are independent of the classical number-theoretic assumptions such as the factoring or discrete logarithm function, and that Strong PUFs can help to establish an independent basis for cryptography and security. Furthermore, they have other security advantages, as discussed in the remainder of this section. The only subtype of Strong PUFs whose security is strictly independent of computational assumptions are SHIC PUFs [10, 97, 98]. The price they pay for this feature is an intrinsically slow readout speed and a comparably large area consumption.

This brings us to another central point related to Strong PUF security. Strong PUFs avoid the use of explicit digital keys in hardware. But do they avoid the presence of secret information in hardware in general? Once the internal configuration of a Strong PUF has become known, an adversary will almost always be able to predict and hence break the PUF. To illustrate our point, consider the Arbiter PUF and its variants: Once the internal runtime delays have become known, the structure can be fully predicted and broken. Therefore Strong PUFs, just like classical cryptosystems, often depend on the assumption that some internal information remains secret. In their favor, this information is arguably hidden better than if stored explicitly in the form of a digital key. F will usually be known to the adversary and efficiently computable.

Security benchmarks. Natural security benchmarks for Strong PUFs must evaluate the complexity of their challenge-response behavior and their resilience against modeling attacks. To this end, various measures have been proposed: (i) Theoretical analysis of the overall internal entropy of the PUF [69]. (ii) Theoretical analysis of the entropy / information-theoretic independence of the CRPs [99–101]. (iii) Empirical, statistical analysis of large CRP sets by statistical tools and compression algorithms [95, 102, 103]. (iv) Empirical analysis by assessment of machine learning curves over instances of increasing size and complexity [95, 103].

Let us briefly discuss these approaches. One downside of (i) is that it usually does not consider the CRP-relevant entropy, but the general entropy of the system, which is often very much larger. (ii) is a suitable measure. On the downside, it can be difficult to derive theoretically, and does not take into account computational aspects. (iii) and (iv) are easy to apply and generic tools, but do not provide definite security guarantees. (iii) does not require an generic model of the PUF (such as the linear additive delay model for arbiter PUFs), while method (iv) needs such a model before it can be applied.

5 Controlled Physical Unclonable Functions (CPUFs)

5.1 Specification of Controlled PUFs

Let us start by specifying the notion of a Controlled PUF: A Controlled Physical Unclonable Function (CPUF) is a PUF that has been bound with an algorithm in such a way that it can only be accessed through a specific Application Programming Interface (API).

The main problem with (uncontrolled) Strong PUFs is that anybody can query the PUF for the response to any challenge. To engage in cryptography with a PUF device, a user who knows a CRP has to use the fact that only he and the device know the response to the user's challenge. But to exploit that fact, the user has to tell the device his challenge so that it can get the response. The challenge has to be told in the clear because there is no key yet. Thus a man in the middle can hear the challenge, get the response from the PUF device and use it to spoof the PUF device.

Clearly the problem in this attack is that the adversary can freely query the PUF to get the response to the user's challenge. By using a CPUF in which access to the PUF is restricted by a control algorithm, this attack can be prevented. The API through which the PUF is accessed should prevent the man-in-the-middle attack we have described without imposing unnecessary limitations on applications.

5.2 History and Implementation

CPUFs can perform all operations that a Strong PUF can perform. While the details of various CPUF APIs are beyond the scope of this paper, useful APIs have been developed [70, 104] that satisfy the following properties:

1. *Access Control.* Anybody who knows a CRP that nobody else knows, can interact with the CPUF device to obtain an arbitrary number of other CRPs that nobody else knows. Thus users are not limited to using a small number of digital outputs from the PUF. Moreover, if one of these new CRPs was revealed to an adversary, transactions that use the other CRPs are not compromised. This is analogous to key management schemes that use session keys derived from a master key.
2. *Secret Sharing.* Anybody can use a CRP that only they know to establish a shared secret with the PUF device. Having a shared secret with the PUF device enables a wide variety of standard cryptographic primitives to be used.
3. *Control Algorithm.* The control algorithm is deterministic. Since hardware random number generators are sensitive and prone to attack, being able to avoid them is advantageous.
4. *Cryptographic Primitive.* The only cryptographic primitive that needs to be built into the control algorithm is a collision resistant hash function. All other cryptographic primitives can be updated during the lifetime of the CPUF device.

By selecting an appropriate API, a CPUF device can be resistant to protocol attacks. With careful design, Optical and Silicon PUFs can be made in such a way that the chip containing the control logic is physically embedded within the PUF: the chip can be embedded within the bubble-containing medium of an Optical PUF, or the delay wires of a Silicon PUF can form a cage on the top chip layer. This embedding should make probing of the control logic considerably more difficult, as an invasive attacker will have to access the wires to be probed without changing the response of the surrounding PUF medium.

The PUF and its control logic have complementary roles. The PUF protects the control logic from invasive attacks, while the control logic protects the PUF from protocol attacks. This synergy makes a CPUF far more secure than either the PUF or the control logic taken independently. Figure 13 demonstrates an example architecture of how a controlled PUF can be used for improving a PUF. A random hash function is placed before the PUF to prevent the adversary from doing a PUF chosen challenge attack. So a model-

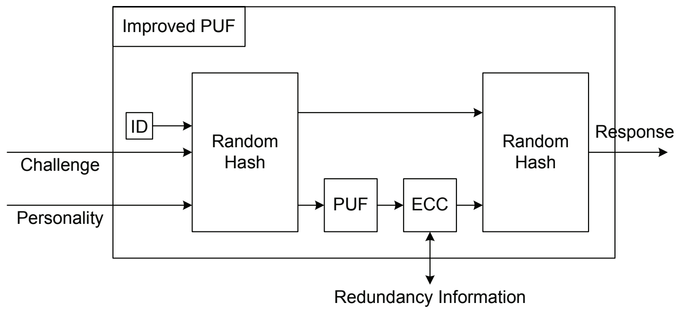


Fig. 13. An example architecture for a controlled PUF proposed in [70].

building adversary is prevented from selecting challenges that allow him to extract the PUF parameters. To ensure response consistency, an Error Correcting Code (ECC) is used. An output random hash function is used to decorrelate the response from the actual physical measurements, and therefore rendering a model-building adversary's task even harder.

5.3 Protocols, Applications, and Security

Because there is no algorithmic way to tie together all the keys produced by a device, the device will have to take an active part in protocols like certificate verification, that would not usually need any device involvement. This limitation is offset by a decreased vulnerability to invasive attacks.

There are many applications for which CPUFs can be used, and we give two examples here. Other applications can be imagined by studying the literature on secure coprocessors, in particular [105]. We note that the general applications for which this technology can be used include all the applications today in which there is a single symmetric key on a chip.

A bank could use certified execution to authenticate messages from PUF smartcards. This guarantees that the message the bank receives originated from the smartcard. It does not, however authenticate the bearer of the smartcard. Some other means such as a PIN number or biometrics must be used by the smartcard to determine if its bearer is allowed to use it. If the privacy of the smartcard's message is a requirement, then the message can also be encrypted.

A second application is for computers that implement private storage [106–112]. A program wishing to store encrypted data in untrusted memory uses an encryption key which depends uniquely on the PUF and its program hash. This requires a CPUF in order to accomplish the unique dependency. This idea is implemented in the AEGIS processor [112, 113].

Physically obfuscated keys generated from Weak PUFs seem to increase the difficulty of an invasive attack, but they still have a single digital point of failure. When the device is in use, the single physically obfuscated master key is present on it in digital form. If an adversary can get that key he has totally broken the device's security. CPUFs exploit the parameterizability of the complex physical system like Strong PUFs do. For each input to the physical system, a different key is produced. Thus the complexity of the physical system is exploited to the utmost.

As noted previously one difficulty with Weak PUFs is that their output is noisy. For use in cryptography, we need error-correction which does not compromise the security is required. For Weak PUFs only one response has to be made noise-free, for CPUFs many responses have to potentially be corrected. We need to store an error correcting syndrome with each challenge-response pair. Secure and robust error correction has been considered for Weak PUFs (see [7]) but these schemes need to be efficiently generalized to CPUFs.

6 Emerging PUF Concepts

There are a number of new concepts that have emerged in the area of PUFs, and the pace of innovation is rapid. We mention interesting new concepts proposed in the past couple of years in this section, and address ongoing research challenges in Section 7.

6.1 PUFs with Secret Models

In classical identification schemes based on Strong PUFs, the verifier must possess a large list of CRPs that have been pre-measured in a secure bootstrapping phase [68, 1]. The challenges sent to the prover are chosen randomly from this list, and the responses obtained from the prover are verified for correctness against this list. Since the list must suffice for the lifetime of the device, it must be large, which imposes uncomfortable storage requirements on the verifier.

It has been independently observed by [114, 115, 77] that such storage requirements may be lifted if the verifier instead stores a secret model for the PUF, by which he can simulate and predict arbitrary responses of the PUF. Such secret models can furthermore allow the offline verification of a PUF's identity, i.e., they can enable identification protocols that are run without an online connection to a trusted authority holding a CRP-list. The underlying PUF primitive could be called *Secret Model PUF* or *SM PUF*, for short.

Secret Model PUFs are a very useful concept that leads to improved practicality features and new protocols. They do not lift two important constraints of Strong PUFs, though: First, the model itself must be kept secret, similar to a secret key. They therefore require the authenticating entity to store a symmetric key to decrypt the secret model stored in encrypted form on the PUF device. Second, SM PUFs still contain some secret information, namely the information that was used to set up the secret model (for example the internal runtime delays). These two requirements are only overcome by the concepts proposed in the next subsections 6.2 and 6.3.

6.2 Timed Authentication

For certain implementations of Strong PUFs, the real-time interval in which the PUF generates its responses may be noticeably shorter than the time that any numerical model or purported clone would require to the same end.

In a PUF-related context, this observation has first been stated in [77]. They noted that for certain FPGA-based PUFs, only the authentic hardware would be able to generate the response in a minimum number of cycles, and that a model built based on the device characteristics would likely be slower in finding the response to a given challenge (compared to the original device). They proposed an authentication protocol that exploits this unique property of the original FPGA device: A time-bound set by the protocol for obtaining the correct response after applying a random challenge ensured that only the authentic device could respond. This scheme has been referred to as Timed Authentication (TA) in [77].

[77] suggests an "asymmetry" in the timed computational capabilities of the authentic device compared to other entities. This asymmetry was elaborated on for thwarting the modeling attacks. However, the proposed protocol is a symmetric key like scheme, since it requires a secret list of CRPs. It was noted that asymmetry can lift the feature that the internal configuration of the PUF-hardware must remain secret. Think of the optical PUF introduced in Section 4.1 as an example: Even if the position of all internal scattering elements/bubbles was known to an adversary, he would still find it hard to simulate the complex input-output behavior of the scattering medium in real-time. The same holds for the FPGA-based implementation of TA discussed in [77].

6.3 PUFs with Public Models

Section 6.1 told us that a secret model for a Strong PUF can replace the CRP list. Section 6.2 described that certain Strong PUFs operate faster than any adversarial model and emulation. Both concepts can be combined to enable PUFs with simulation models that can be made public (and hence can be simulated by everyone), but which still operate faster than any clone or model (including the public model, of course). The manufacturer or some other entity can tie the model to the respective PUF, by, for example, signing the model, or keeping it in a trusted public register. This allows everyone to simulate the responses of the PUF with some time overhead. Only the party holding the PUF can determine the PUF's responses fast, i.e., within a certain time bound, by a physical measurement on the PUF. This allows public key like functionalities and protocols. Hardware systems based on such a concept have the further intriguing advantage that they can eradicate the presence of any form of secret information in the cryptographic hardware, while still being usable in typical digital network applications such as remote identification and message authentication.

History. The concept of PUFs with public models has been introduced and developed independently in several lines of research. Under the name of a Public PUF (PPUF), this primitive has been introduced in [116–118], building on a hardware concept that had been published earlier [119, 120, 81, 77]. Protocols and applications of PPUFs have since been developed [117, 118, 121]. Under the name of a SIMPL system, the same concept was put forward completely independently in [122, 123]. Implementations, protocols and applications of SIMPLs have been elaborated on in [124–127, 90, 128]. In another line of research, the concept of PUFs with public models has been made explicit with implementation results on FPGAs under the name Time-Bounded Authentication (TBA) in [92, 9]; this builds on the concept of TA treated in the last section [77].

6.4 Quantum Readout PUFs

[129] proposed modifying the challenge-response mechanism of a PUF with quantum states, called a *Quantum Readout PUF* [130]. The properties of the quantum states prevent an adversary from intercepting the challenges and responses without modifying them. Thus, there is no need for a trusted location for bootstrapping. However, no proof-of-concept implementation or practical architecture for this structure has been proposed to date. Finally, interfacing the quantum readout device to the regular PUF is likely a challenge.

6.5 SHIC PUFs

A final recent concept are PUFs with Super-High Information Content, abbreviated *SHIC PUFs* [1] [10, 97, 98]. SHIC PUFs are Strong PUFs whose large number of CRPs are pairwise independent in an information-theoretic sense. Unlike other Strong PUFs, this allows them to become independent of computational assumptions in their security. The price they pay is a relatively large area consumption and slow read-out speed on the order of 10^2 to 10^4 bits per second. SHIC PUFs are unlikely to be used in low-cost commercial applications in the near future, since there are other, more favorable solutions to this end. But they represent an intriguing theoretical tool, since they are a variant of Strong PUFs with information-theoretic security. Furthermore, investigating their optimal implementation is rewarding from a technological perspective, since it relates to fundamental technological questions such as "How much random information can we store and reliably extract from a solid-state system?",

[1] SHIC PUFs are to be pronounced as *"chique PUFs"* according to [10].

and “How can we make the speed in which information is released from a solid-state system inherently *slow*?”.

7 Future Research Topics

7.1 Open Public PUF Questions

The main open questions related to PUFs with Public Models concern their hardware realization:

- How can it be guaranteed that the model requires more time to simulate than the PUF device requires to return a response?
- How can it be guaranteed that a well-equipped adversary for sure takes longer than the PUF device, while any poorly equipped honest party can simulate the response in feasible time in the course of a communication protocol?
- Can the model be close enough to the PUF so that an adversary finds it difficult to physically clone the PUF, but loose enough to allow for variation due to environmental conditions of PUF responses?

While there have been many recent proposals for timed authentication, we are not aware of any implementation that definitively settles the above questions. This leaves strong potential for future research. If a workable, small and inexpensive implementation of PPUFs, SIMPL systems or TBA systems is found eventually, or if one of the existing implementations is shown to possess all necessary properties, this would have a massive impact on the way we perform cryptography and construct security hardware.

7.2 Efficient Hardware Implementations: Overhead versus Security

Recent work has discussed how it could be possible to safeguard PUFs against reverse-engineering and modeling attacks [77, 54, 88, 10, 97, 98]. However, most methods that aim at protecting against such attacks add strongly to the power, size, delay, instability, or cost overhead of the system. Also techniques for ensuring the tamper-proof properties, such as inaccessibility of the Weak PUF, would require addition of tamper-proof circuitry and material to the devices.

One major future research topic is how the security of Strong PUFs and/or the tamper sensitivity of Strong PUFs and Weak PUFs can be realized with a minimal hardware overhead. These future research questions naturally relate to circuit design and, concerning tamper sensitivity, also to the material sciences.

7.3 Error Correction and Practical Operability

A suite of security applications of PUFs, such as secret key generation by Weak PUFs, require full and error-free reconstruction of the keys. However, environmental conditions and aging may affect the measured responses in strong ways. Methods for compensation of such effects, such as circuit reliability enhancement techniques, error correction and secure sketches, are being developed [2, 131, 8, 4, 5, 7]. Further development of methods that ensure robustness of PUFs with a limited amount of leaked information is of great interest. One key challenge is that the maximum number of unpredictable bits should be known at the design time. If the unpredictability exceeds the bound set at the time of design, the error correction method would not be able to compensate for the errors. Therefore, careful experimental studies for each new PUF structure are needed for characterizing the performance under different temperature, voltage, and/or other environmental and operational conditions, constituting a future area of active and fruitful interplay between hardware analysis and error correction techniques.

7.4 IC Metering and Counterfeit Detection

A counterfeit product is an illegal forgery or imitation of an original design. Because of the dominance of the contract foundry model, IP sharing/reuse, and outsourcing, the electronic products are increasingly vulnerable to piracy attack and counterfeiting. *IC metering* is a set of security protocols that enable the design house (authentic IP owner) to achieve post-fabrication control over their ICs [119, 132, 66, 133]. In *passive IC metering*, the IP rights owner is able to identify and monitor the devices [119, 132]. Passive metering can be directly enabled by certain types of PUFs. In *active IP metering*, in addition to identification and monitoring, the IP rights holder can actively control, enable/disable, and authenticate a device [66]. We refer the interested readers to Chapter 8 of this book for a comprehensive survey of this topic. Addressing piracy attacks is notoriously hard since the adversaries are often financially strong, technologically advanced and informed of the design details. A set of open research questions have to do with developing security methods, PUF architectures, and controlled PUF protocols that can directly address the piracy attack models and counterfeiting.

7.5 Attacks and Vulnerability Analysis

To date, a number of attacks and countermeasures for PUFs are reported, see for example the detailed discussions in Section 4.2. However, PUFs have yet to undergo more refined cryptanalysis and evaluation of physical and side-channel attacks by a large community of researchers, similar to the way many traditional cryptographic primitives and protocols have been analyzed and attacked. For PUFs to be widely accepted, this seems to be a central future task that needs to be performed.

7.6 Formalization and Security Proofs

One relatively untouched area within physical cryptography and PUFs are the foundations of these fields. Formal definitions and security proofs for PUF-based protocols are just about to develop. For example, [71, 72] provide a thorough discussion of existing PUF definitions. [72] give new formal definitions for Strong PUFs that lead to a first reductionist security proof for a Strong PUF-based identification scheme. This type of work will likely prove essential for sound future development of the field, and will represent one of the major upcoming research topics within the area.

7.7 New Protocols and Applications

Up to now, PUFs and UNOs have mainly been used for authentication and identification purposes, and have mainly been seen as a security tool. But recently, a fundamental result indicated that PUFs possess a strong cryptographic potential: Oblivious transfer (and all protocols that can be derived from it) can be realized by

Strong PUFs [134]. Protocol design and optimization will thus be active future research topics.

8 Conclusion

Security and protection based on random physical media and objects is a fast-growing field that has recently enjoyed considerable research interest. Ensuring authenticity, security, protection, and integrity of data, hardware and software intellectual property, computers, networks, identities, and cyber-physical systems is a standing challenge. Traditional digital methods for these tasks often rely on digital labels or digitally stored secret keys that are vulnerable to forging, cloning, and other attacks. As discussed extensively in the previous sections, the unique and unclonable character of disordered physical structures can be exploited to address many of the vulnerabilities of these traditional concepts.

This chapter presented a new classification for the area of physical disorder based cryptography and security. We dealt with disorder-based identification, authentication, and other security methods. We then focused on four new classes of security devices based on physical disorder: Unique Objects, Weak Physical Unclonable Functions (Weak PUFs), Strong PUFs, and Controlled PUFs. Alongside with defining each class and discussing the history and relevant work, we described existing hardware implementations of these novel security primitives. We discussed emerging concepts in the area, including Timed Authentication and Public PUFs and SIMPL systems. We complemented the chapter by a treatment of future research challenges, which could prove helpful as a guideline to graduate students or anyone who wants to conduct research in the area.

9 Acknowledgement

The authors would like to thank Prof. Wayne P. Burleson for his valuable comments and suggestions. The authors would also like to thank Azalia Mirhoseini for her help with some of the figures.

References

1. B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, “Silicon physical random functions,” in Computer and Communication Security Conference, 2002.
2. B. Gassend, “Physical Random Functions,” Master’s thesis, Massachusetts Institute of Technology, Jan. 2003.
3. G. Suh, C. O’Donnell, and S. Devadas, “AEGIS: a Single-Chip secure processor,” IEEE Design & Test of Computers, vol. 24, no. 6, pp. 570–580, 2007.
4. C. Yin and G. Qu, “LISA: maximizing RO PUF’s secret extraction,” in Hardware-Oriented Security and Trust (HOST), 2010, pp. 100–105.
5. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, “Extended abstract: The butterfly PUF protecting IP on every FPGA,” in Hardware-Oriented Security and Trust (HOST), 2008, pp. 67–70.
6. R. Maes, P. Tuyls, and I. Verbauwhede, “Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs,” in Cryptographic Hardware and Embedded Systems (CHES), 2009, pp. 332–347.
7. M.-D. M. Yu and S. Devadas, “Secure and robust error correction for physical unclonable functions,” IEEE Design and Test of Computers, vol. 27, pp. 48–65, 2010.
8. M. Majzoobi, F. Koushanfar, and S. Devadas, “FPGA PUF using programmable delay lines,” in IEEE Workshop on Information Forensics and Security, 2010, p. in press.
9. M. Majzoobi and F. Koushanfar, “Time-Bounded Authentication of FPGAs,” in Under Revision for IEEE Trans. on Information Forensics and Security (TIFS), 2011.
10. U. Rührmair, C. Jaeger, C. Hilgers, M. Algasinger, G. Csaba, and M. Stutzmann, “Security applications of diodes with unique current-voltage characteristics,” Financial Cryptography and Data Security (FC), pp. 328–335, 2010.
11. G. Suh and S. Devadas, “Physical unclonable functions for device authentication and secret key generation,” in Design Automation Conference (DAC), 2007, pp. 9–14.
12. A. Sadeghi and D. Naccache, Eds., Towards Hardware-Intrinsic Security: Foundations and Practice. Springer, 2010.
13. D. Kirovski, “Anti-Counterfeiting: Mixing the Physical and the Digital World,” in Towards Hardware-Intrinsic Security, A.-R. Sadeghi and D. Naccache, Eds. Springer, 2010, pp. 223–233.
14. S. Li and A. Jain, Eds., Encyclopedia of Biometrics. Springer, 2009.
15. D. Maltoni, D. Maio, A. Jain, and S. Prabhakar, Handbook of Fingerprint Recognition. Springer, 2009.
16. D. Kirovski, personal communication, Dagstuhl, Germany, 2008.
17. S. Graybeal and P. McFate, “Getting out of the STARTing block,” Scientific American (USA), vol. 261, no. 6, 1989.
18. D. Bauder, “An anti-counterfeiting concept for currency systems,” Research report PTK-11990. Sandia National Labs. Albuquerque, NM, 1983.
19. J. Brosow and E. Furugard, “Method and a system for verifying authenticity safe against forgery,” US Patent 4,218,674, 1980.
20. G. Simmons, “A system for verifying user identity and authorization at the point-of sale or access,” Cryptologia, vol. 8, no. 1, pp. 1–21, 1984.
21. ——, “Identification of data, devices, documents and individuals,” in IEEE International Carnahan Conference on Security Technology, 1991, pp. 197–218.

22. J. Buchanan, R. Cowburn, A. Jausovec, D. Petit, P. Seem, G. Xiong, D. Atkinson, K. Fenton, D. Allwood, and M. Bryan, "Forgery:fingerprintingdocuments and packaging," Nature, vol. 436, no. 7050, p. 475, 2005.
23. J. Smith and A. Sutherland, "Microstructure based indicia," Proceedings of the Automatic Identification Advanced Technologies AutoID, vol. 99, pp. 79–83, 1999.
24. E. Métois, P. Yarin, N. Salzman, and J. Smith, "FiberFingerprint identification," in Workshop on Automatic Identification, 2002, pp. 147–154.
25. P. Seem, J. Buchanan, and R. Cowburn, "Impact of surface roughness on laser surface authentication signatures under linear and rotational displacements," Optics letters, vol. 34, no. 20, pp. 3175–3177, 2009.
26. A. Sharma, L. Subramanian, and E. Brewer, "Secure rural supply chain management using low cost paper watermarking," in ACM SIGCOMM workshop on Networked systems for developing regions, 2008, pp. 19–24.
27. F. Beekhof, S. Voloshynovskiy, O. Koval, R. Villan, and T. Pun, "Secure surface identification codes," in Proceedings of SPIE, vol. 6819, 2008, p. 68190D.
28. W. Clarkson, T. Weyrich, A. Finkelstein, N. Heninger, J. Halderman, and E. Felten, "Fingerprinting blank paper using commodity scanners," in IEEE Symposium on Security and Privacy, 2009, pp. 301–314.
29. The ProteXXion System, Bayer AG, http://www.research.bayer.com/edition-19/protexxion.aspx and http://www.research.bayer.com/edition-19/19_Protexxion_en.pdfx.
30. Ingeniatechnology, http://www.ingeniatechnology.com/.
31. G. DeJean and D. Kirovski, "RF-DNA: Radio-frequency certificates of authenticity," Cryptographic Hardware and Embedded Systems (CHES), pp. 346–363, 2007.
32. D. Kirovski, "Toward an automated verification of certificates of authenticity," in ACM Electronic Commerce (EC), 2004, pp. 160–169.
33. Y. Chen, M. Mihçak, and D. Kirovski, "Certifying authenticity via fiber-infused paper," ACM SIGecom Exchanges, vol. 5, no. 3, pp. 29–37, 2005.
34. P. Bulens, F. Standaert, and J. Quisquater, "How to strongly link data and its medium: the paper case," IET Information Security, vol. 4, no. 3, pp. 125–136, 2010.
35. Y. Kariakin, "Authentication of articles," Patent writing, WO/1997/024699, available from http://www.wipo.int/pctdb/en/wo.jsp?wo=1997024699, 1995.
36. G. Hammouri, A. Dana, and B. Sunar, "CDs have fingerprints too," Cryptographic Hardware and Embedded Systems (CHES), pp. 348–362, 2009.
37. D. Vijaywargi, D. Lewis, and D. Kirovski, "Optical DNA," Financial Cryptography and Data Security (FC), pp. 222–229, 2009.
38. B. Zhu, J. Wu, and M. Kankanhalli, "Print signatures for document authentication," in Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS). ACM, 2003, pp. 145–154.
39. J. Collins, "RFID Fibers for Secure Applications," RFID Journal, vol. 26, 2004.
40. RF SAW Inc., http://www.rfsaw.com/tech.html.
41. Creo Inc., http://www.creo.com.
42. Inkode Inc., http://www.inkode.com.
43. Microtag Temed Ltd, http://www.microtag-temed.com/.
44. CrossID Inc., Firewall Protection for Paper Documents, http://www.rfidjournal.com/article/articleview/790/1/44.
45. C. Loibl, "Entwurf und Untersuchung berührungslos abfragbarer einzigartiger Objekte," Master's thesis, Fachgebiet Höchstfrequenztechnik, Technische Universität München, 2009.

46. MagnePrint, http://www.magneprint.com/.
47. U. Rührmair, M. Stutzmann, P. Lugli, C. Jirauschek, K. Müller, H. Langhuth, G. Csaba, E. Biebl, and J. Finley, "Method and system for security purposes," European Patent Application Nr. EP 09 157 041.6, March 2009.
48. C. Clelland, V. Risca, and C. Bancroft, "Hiding messages in DNA microdots," Nature, vol. 399, no. 6736, pp. 533–534, 1999.
49. November AG, http://www.november.de/archiv/pressemitteilungen/pressemitteilung/article/sichere-medikamente-dank-dna-codes-der-identif-gmbh.html.
50. D. Kirovski, "A point-set compression heuristic for fiber-based certificates of authenticity," in Data Compression Conference (DCC), 2005, pp. 103–112.
51. ——, "Point compression for certificates of authenticity," in Data Compression Conference (DCC), 2004, p. 545.
52. Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in Advances in cryptology-Eurocrypt 2004. Springer, 2004, pp. 523–540.
53. Alliance for Gray Market and Counterfeit Abatement (AGMA), http://www.agmaglobal.org/.
54. U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in ACM Conference on Computer and Communications Security (CCS), 2010, pp. 237–249.
55. C. Bennett, G. Brassard, S. Breidbart, and S. Wiesner, "Quantum cryptography, or unforgeable subway tokens," in Advances in Cryptology–Proceedings of Crypto, vol. 82, 1983, pp. 267–275.
56. C. Bennett, G. Brassard, et al., "Quantum cryptography: Public key distribution and coin tossing," in International Conference on Computers, Systems and Signal Processing, vol. 175. Bangalore, India, 1984.
57. J. Guajardo, S. Kumar, G. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in Cryptographic Hardware and Embedded Systems (CHES), 2007, pp. 63–80.
58. K. Lofstrom, W. R. Daasch, and D. Taylor, "Ic identification circuit using device mismatch," in ISSCC, 2000, pp. 372–373.
59. P. Layman, S. Chaudhry, J. Norman, and J. Thomson, "Electronic fingerprinting of semiconductor integrated circuits," US Patent 6,738,294, September 2002.
60. Y. Su, J. Holleman, and B. Otis, "A 1.6pJ/bit 96 (percent) stable chip ID generating circuit using process variations," in IEEE International Solid-State Circuits Conference (ISSCC), 2007, pp. 200–201.
61. D. Holcomb, W. Burleson, and K. Fu, "Initial SRAM state as a fingerprint and source of true random numbers for RFID tags," in Proceedings of the Conference on RFID Security, 2007.
62. P. Tuyls, G.-J. Schrijen, B. Skoric, J. van Geloven, N. Verhaegh, and R. Wolters, "Read-proof hardware from protective coatings," in Cryptographic Hardware and Embedded Systems (CHES), 2006, pp. 369–383.
63. R. Helinski, D. Acharyya, and J. Plusquellic, "A physical unclonable function defined using power distribution system equivalent resistance variations," in Design Automation Conference (DAC), 2009, pp. 676–681.
64. ——, "Quality metric evaluation of a physical unclonable function derived from an IC's power distribution system," in Design Automation Conference, ser. DAC, 2010, pp. 240–243.

65. G. E. Suh, “AEGIS: A Single-Chip Secure Processor,” Ph.D. dissertation, Massachusetts Institute of Technology, Aug 2005.
66. Y. Alkabani and F. Koushanfar, “Active hardware metering for intellectual property protection and security,” in USENIX Security Symposium, 2007, pp. 291–306.
67. D. Holcomb, W. Burleson, and K. Fu, “Power-up SRAM state as an identifying fingerprint and source of true random numbers,” IEEE Transactions on Computers, vol. 58, no. 9, pp. 1198–1210, September 2009.
68. R. Pappu, “Physical one-way functions,” Ph.D. dissertation, Massachusetts Institute of Technology, 2001.
69. R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, “Physical one-way functions,” Science, vol. 297, pp. 2026–2030, 2002.
70. B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, “ Controlled Physical Random Functions ,” in Annual Computer Security Applications Conference, 2002.
71. U. Rührmair, F. Sehnke, and J. Sölter, “On the Foundations of Physical Unclonable Functions,” Cryptology ePrint Archive, International Association for Cryptologic Research, Tech. Rep., 2009.
72. U. Rührmair, H. Busch, and S. Katzenbeisser, “Strong PUFs: Models, Constructions, and Security Proofs,” in Towards Hardware-Intrinsic Security, A.-R. Sadeghi and D. Naccache, Eds. Springer, 2010, pp. 79–96.
73. B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, “Delay-Based Circuit Authentication and Applications,” in Symposium on Applied Computing (SAC), 2003.
74. J.-W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, “A technique to build a secret key in integrated circuits with identification and authentication applications,” in IEEE VLSI Circuits Symposium, New-York, June 2004.
75. D. Lim, “Extracting Secret Keys from Integrated Circuits,” Master’s thesis, Massachusetts Institute of Technology, may 2004.
76. B. Gassend, D. Lim, D. Clarke, M. van Dijk, and S. Devadas, “Identification and authentication of integrated circuits,” Concurrency and Computation: Practice and Experience, vol. 16, no. 11, pp. 1077–1098, 2004.
77. M. Majzoobi, F. Koushanfar, and M. Potkonjak, “Techniques for design and implementation of secure reconfigurable pufs,” ACM Transactions on Reconfigurable Technology and Systems (TRETS), vol. 2, no. 1, 2009.
78. S. Devadas, E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal, “Design and Implementation of PUF-Based ”Unclonable” RFID ICs for Anti-Counterfeiting and Security Applications,” in Proceedings of 2008 IEEE International Conference on RFID (RFID 2008), May 2008, pp. 58–64.
79. D. Suzuki and K. Shimizu, “The Glitch PUF: A New Delay-PUF Architecture Exploiting Glitch Shapes,” Cryptographic Hardware and Embedded Systems (CHES), 2010, pp. 366–382.
80. S. Devadas and B. Gassend, “Authentication of integrated circuits,” US Patent 7,840,803, 2010, application in 2002.
81. Y. Alkabani, F. Koushanfar, N. Kiyavash, and M. Potkonjak, “Trusted integrated circuits: A nondestructive hidden characteristics extraction approach,” in Information Hiding (IH), 2008, pp. 102–117.
82. M. Potkonjak and F. Koushanfar, “Identification of integrated circuits,” US Patent Application 12/463,984; Publication Number: US 2010/0287604 A1, May 2009.

83. F. Koushanfar, P. Boufounos, and D. Shamsi, "Post-silicon timing characterization by compressed sensing," in International Conference on Computer-Aided Design (ICCAD), 2008, pp. 185–189.
84. D. Shamsi, P. Boufounos, and F. Koushanfar, "Noninvasive leakage power tomography of integrated circuits by compressive sensing," in International Symposium on Low Power Electronic Designs (ISLPED), 2008, pp. 341–346.
85. M. Nelson, A. Nahapetian, F. Koushanfar, and M. Potkonjak, "Svd-based ghost circuitry detection," in Information Hiding (IH), 2009, pp. 221–234.
86. S. Wei, S. Meguerdichian, and M. Potkonjak, "Gate-level characterization: Foundations and hardware security applications," in Design Automation Conference (DAC), 2010.
87. F. Koushanfar and A. Mirhoseini, "A unified framework for multimodal submodular integrated circuits trojan detection," IEEE Trans. on Information Forensic and Security (TIFS), 2011.
88. G. Csaba, X. Ju, Z. Ma, Q. Chen, W. Porod, J. Schmidhuber, U. Schlichtmann, P. Lugli, and U. Ruhrmair, "Application of mismatched cellular nonlinear networks for physical cryptography," in International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA). IEEE, 2010, pp. 1–6.
89. P. Tuyls and B. Škorić, "Strong Authentication with Physical Unclonable Functions," Security, Privacy, and Trust in Modern Data Management, pp. 133–148, 2007.
90. U. Rührmair, "SIMPL Systems, Or: Can we construct cryptographic hardware without secret key information?" in International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM), ser. Lecture Notes in Computer Science, vol. 6543. Springer, 2011.
91. U. Rührmair, C. Jaeger, and M. Algasinger, "An Attack on PUF-based Session Key Exchange and a Hardware-based Countermeasure," Financial Cryptography and Data Security (FC), 2011, to appear.
92. M. Majzoobi, A. E. Nably, and F. Koushanfar, "FPGA Time-Bounded Authentication," in Information Hiding Conference (IH), 2010, pp. 1–15.
93. J. Bekenstein, "How does the entropy/information bound work?" Foundations of Physics, vol. 35, no. 11, pp. 1805–1823, 2005.
94. E. Oztürk, G. Hammouri, and B. Sunar, "Towards robust low cost authentication for pervasive devices," in Pervasive Computing and Communications (PerCom), 2008, pp. 170–178.
95. M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing techniques for hardware security," in International Test Conference (ITC), 2008, pp. 1–10.
96. ——, "Lightweight secure PUF," in International Conference on Computer Aided Design (ICCAD), 2008, pp. 670–673.
97. U. Rührmair, C. Jaeger, M. Bator, M. Stutzmann, P. Lugli, and G. Csaba, "Applications of high-capacity crossbar memories in cryptography," IEEE Transactions on Nanotechnology, no. 99, p. 1.
98. C. Jaeger, M. Algasinger, U. Rührmair, G. Csaba, and M. Stutzmann, "Random pn-junctions for physical cryptography," Applied Physics Letters, vol. 96, p. 172103, 2010.
99. P. Tuyls, B. Skoric, S. Stallinga, A. H. M. Akkermans, and W. Ophey, "Information-theoretic security analysis of physical uncloneable functions," in Financial Cryptography and Data Security (FC), 2005, pp. 141–155.
100. B. Škorić, "On the entropy of keys derived from laser speckle; statistical properties of Gabor-transformed speckle," Journal of Optics A: Pure and Applied Optics, vol. 10, p. 055304, 2008.

101. B. Skoric, S. Maubach, T. Kevenaar, and P. Tuyls, "Information-theoretic analysis of capacitive physical unclonable functions," Journal of Applied Physics, vol. 100, no. 2, p. 024902, 2009.
102. I. Kim, A. Maiti, L. Nazhandali, P. Schaumont, V. Vivekraja, and H. Zhang, "From Statistics to Circuits: Foundations for Future Physical Unclonable Functions," Towards Hardware-Intrinsic Security, pp. 55–78, 2010.
103. F. Sehnke, J. Schmidhuber, and U. Rührmair, "Security Benchmarks for Strong Physical Unclonable Functions," 2010, in submission.
104. B. Gassend, M. van Dijk, D. Clarke, E. Torlak, S. Devadas, and P. Tuyls, "Controlled physical random functions and applications," ACM Transactions on Information and System Security (TISSEC), vol. 10, no. 4, pp. 1–22, 2008.
105. B. S. Yee, "Using secure coprocessors," Ph.D. dissertation, Carnegie Mellon University, 1994.
106. A. Carroll, M. Juarez, J. Polk, and T. Leininger, "Microsoft "palladium": A business overview," in Microsoft Content Security Business Unit, August 2002. [Online]. Available: http://www.microsoft.com/presspass/features/2002/jul02/0724palladiumwp.asp
107. T. Alves and D. Felton, "Trustzone: Integrated hardware and software security," ARM white paper, jul 2004.
108. Microsoft, "Next-Generation Secure Computing Base," http://www.microsoft.com/ `resources/ngscb/defaul.mspx`.
109. T. C. Group, "Tcg specification architecture overview revision 1.2," http://www.trustedcomputinggroup.com/home, 2004.
110. D. Lie, C. Thekkath, M. Mitchell, P. Lincoln, D. Boneh, J. Mitchell, and M. Horowitz, "Architectural support for copy and tamper resistant software," in Int'l Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX), 2000, pp. 168–177.
111. D. Lie, "Architectural support for copy and tamper-resistant software," Ph.D. dissertation, Stanford University, Dec 2003.
112. G. E. Suh, D. Clarke, B. Gassend, M. van Dijk, and S. Devadas, "AEGIS: Architecture for tamper-evident and tamper-resistant processing," in Int'l Conference on Supercomputing (MIT-CSAIL-CSG-Memo-474 is an updated version), 2003.
113. 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," in International Symposium on Computer Architecture (ISCA), 2005.
114. S. Devadas, "Non-networked rfid puf authentication," US Patent Application 12/623,045, 2008.
115. E. Oztiirk, G. Hammouri, and B. Sunar, "Towards robust low cost authentication for pervasive devices," in International Conference on Pervasive Computing and Communications (PerCom), 2008, pp. 170 –178.
116. N. Beckmann and M. Potkonjak, "Hardware-based public-key cryptography with public physically unclonable functions," in Information Hiding. Springer, 2009, pp. 206–220.
117. M. Potkonjak, "Secure authentication," US Patent Application 12/464,387; Publication Number: US 2010/0293612 A1, May 2009.
118. ——, "Digital signatures," US Patent Application 12/464,384; Publication Number: US 2010/0293384 A1, May 2009.
119. F. Koushanfar, G. Qu, and M. Potkonjak, "Intellectual property metering," in International Workshop on Information Hiding (IHW), 2001, pp. 81–95.

120. F. Koushanfar and M. Potkonjak, “Cad-based security, cryptography, and digital rights management,” in Design Automation Conference (DAC), 2007, pp. 268–269.
121. M. Potkonjak, S. Meguerdichian, and J. Wong, “Trusted sensors and remote sensing,” in IEEE Sensors, 2010, pp. 1–4.
122. U. Rührmair, M. Stutzmann, G. Csaba, U. Schlichtmann, and P. Lugli, “Method for security purposes,” European Patent Filings EP 09003764.9, EP 09003763.1, EP 09157043.2, March 2009.
123. U. Rührmair, “SIMPL Systems: On a Public Key Variant of Physical Unclonable Functions,” Cryptology ePrint Archive, International Association for Cryptologic Research, Tech. Rep., 2009.
124. U. Rührmair, Q. Chen, M. Stutzmann, P. Lugli, U. Schlichtmann, and G. Csaba, “Towards Electrical, Integrated Implementations of SIMPL Systems,” Cryptology ePrint Archive, International Association for Cryptologic Research, Tech. Rep., 2009.
125. Q. Chen, G. Csaba, X. Ju, S. Natarajan, P. Lugli, M. Stutzmann, U. Schlichtmann, and U. Ruhrmair, “Analog circuits for physical cryptography,” in 12th International Symposium on Integrated Circuits (ISIC’09), Singapore, December 14 – 16, 2009. IEEE, 2009/2010, pp. 121–124.
126. U. Rührmair, Q. Chen, M. Stutzmann, P. Lugli, U. Schlichtmann, and G. Csaba, “Towards electrical, integrated implementations of simpl systems,” in Workshop in Information Security Theory and Practice (WISTP), 2010, pp. 277–292.
127. Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, M. Stutzmann, and U. Rührmair, “Circuit-based approaches to SIMPL systems,” Journal of Circuits, Systems, and Computers, vol. 20, pp. 107–123, 2011.
128. U. Rührmair, “SIMPL Systems as a Cryptographic and Security Primitive,” in To be submitted to IEEE Trans. on Information Forensics and Security (TIFS), 2011.
129. B. Škorić, “Quantum Readout of Physical Unclonable Functions,” Progress in Cryptology–AFRICACRYPT 2010, pp. 369–386, 2010.
130. B. koric, “Quantum readout of physical unclonable functions,” in Progress in Cryptology (AFRICACRYPT), ser. Lecture Notes in Computer Science, D. Bernstein and T. Lange, Eds. Springer Berlin / Heidelberg, 2010, vol. 6055, pp. 369–386.
131. C. Bösch, J. Guajardo, A. Sadeghi, J. Shokrollahi, and P. Tuyls, “Efficient helper data key extractor on FPGAs,” in Cryptographic Hardware and Embedded Systems (CHES), 2008, pp. 181–197.
132. F. Koushanfar and G. Qu, “Hardware metering,” in Design Automation Conference (DAC), ser. DAC, 2001, pp. 490–493.
133. Y. Alkabani, F. Koushanfar, and M. Potkonjak, “Remote activation of ICs for piracy prevention and digital right management,” in ICCAD, 2007.
134. U. Rührmair, “Oblivious transfer based on physical unclonable functions (extended abstract),” in TRUST, ser. Lecture Notes in Computer Science, A. Acquisti, S. W. Smith, and A.-R. Sadeghi, Eds., vol. 6101. Springer, 2010, pp. 430–440.

Part II

Physical Unclonable Functions as an Advanced Cryptographic Primitive

Chapter 3

Oblivious Transfer Based on Physical Unclonable Functions

During their early years, PUFs were commonly regarded as a security tool, in particular as a novel key storage method. This second part of the thesis, which comprises of Chapters 3 to 6, presents an extended view: It explores the theoretical and practical potential of PUFs as a novel primitive in advanced cryptographic protocols. The surprising usefulness of PUFs in this context puts them line with other, well-known alternative cryptographic approaches, for example the bounded storage model [38, 3], quantum cryptography [5], noise-based cryptography [12], or also post-quantum cryptography [4].

This Chapter 3 thereby shows for the first time that oblivious transfer (OT) can be realized merely on the basis of so-called Strong PUFs and SHIC PUFs. Notably, the PUF-based OT-protocol we present does not require any other, additional assumptions than the unpredictability and unclonability of the PUF itself. More precisely, in the case of Strong PUFs, our protoocl establishes computational security based on the unpredictability of the PUF, while in the case of SHIC PUF, it even achieves information-theoretic security; details are provided in the upcoming chapter.

It is important to recall in this context that OT is a universal cryptographic primitive which enables a large number of other protocols, as first shown by Kilian [29] in 1988. For this reason, the implementability (or non-implementability) of OT within a new cryptographic model has traditionally been used as a touchstone for the model's potential. For example, the realizability of OT in noise-based crypto [13] and the bounded storage model [21] were early indicators of the strength of these approaches, while the known problems with implementing OT and bit commitment in quantum cryptography [34, 32] marked some first limitations of this technique. The fact that PUF can indeed realize OT, at least in an isolated stand-alone setting, thus serves as a first indicator of a large cryptographic potential of theirs.

The concrete paper we employ in this chapter is:

- U. Rührmair: *Oblivious Transfer Based on Physical Unclonable Functions*. TRUST 2010, Lecture Notes in Computer Science, Vol. 6101, pp. 430-440, Springer, 2010.

According to Google scholar, it has been cited 44 times to this date [26].

As an outlook, let us mention already now how the upcoming Chapters 4 to 6 continue and complement the work of this Chapter 3.

First of all, Chapter 4 contrasts two possible approaches to PUF-security: A theoretical approach on the one hand, which concentrates on asymptotic security guarantees; and a practically motivated method on the other hand, which focuses on the concretely achievable security in practical applications. By the example of an OT-protocol from CRYPTO 2011 by Brzuska, Fischlin, Schröder and Katzenbeisser [6], we demonstrate that both views can substantially differ, and that the latter method may be preferable, at the very least for any practical and commercial PUF applications.

More specifically, we show that the abovementioned protocol from CRYPTO 2011 [6], which is secure in an asymptotic security model, is not secure in practice when being used in connection with optical PUFs. Such use had been explicitly suggested at CRYPTO 2011, however [6]. This illustrates that care must be taken when common asymptotic concepts from theoretical cryptography are transfered to PUFs. The latter are a novel and special primitive of their own also in this respect. Chapter 4 also discusses measures to improve security, and to overcome the practical vulnerabilities of the CRYPTO 2011 protocol. We show that our own OT-protocol from Chapter 3 has better concrete security guarantees than the protocol from Crypto 2011 [6], while requiring more rounds of communication. Both facts result from the use of interactive hashing as a substep in our protocol.

The subsequent Chapters 5 and 6 continue to study the practical security and applicability of PUF-protocols. They partly relativize too high hopes on the actual effectiveness of PUFs in cryptographic protocols, reporting certain limitations. For example, they illustrate that plain Strong PUF cannot directly be re-used in two consecutive OT-protocols without endangering the security of the first OT. They also show that malicious PUF-manufacturers can influence the security of protocols to their favor by generating biased PUF hardware, so-called *"bad PUFs"*. The same holds not only for malicious manufacturers, but also for malicious players who exchange normal, *"good PUFs"* against such bad PUFs.

Concluding this outlook, we confirm nevertheless that as long as only stand-alone scenarios with a one-time use of "good" and non-manipulated PUFs are considered, the claims that PUFs can realize OT and may serve a universal cryptographic primitive are maintained, as put forward in this Chapter 3.

Oblivious Transfer based on Physical Unclonable Functions (Extended Abstract)

Ulrich Rührmair

Computer Science Department
Technische Universität München
85748 Garching, Germany
ruehrmai@in.tum.de
http://www.pcp.in.tum.de

Abstract. Oblivious transfer (OT) is a simple, but powerful cryptographic primitive, on the basis of which secure two-party computation and several other cryptographic protocols can be realized. In this paper, we show how OT can be implemented by Strong Physical Unclonable Functions (PUFs). Special attention is thereby devoted to a recent subclass of Strong PUFs known as SHIC PUFs. Our results show that the cryptographic potential of these PUFs is perhaps surprisingly large, and goes beyond the usual identification and key exchange protocols.

1 Introduction

Motivation and Background. Electronic devices are becoming increasingly mobile, cross-linked and pervasive, which makes them a well-accessible target for adversaries. Mathematical cryptography offers several measures against the resulting security and privacy problems, but they all rest on the concept of a secret binary key: They presuppose that the devices can contain a piece of information that is, and remains, unknown to an adversary. This requirement can be difficult to uphold in practice: Invasive, semi-invasive, or side-channel attacks, as well as various software attacks including viruses, can lead to key exposure and full security breaks.

The described situation was one motivation that led to the development of *Physical Unclonable Functions (PUFs)* [1]. 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 R_i. Contrary to standard digital systems, a PUF's responses shall depend on the nanoscale structural disorder present in it. 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.

Due to their complex internal structure, PUFs can often avoid some of the shortcomings associated with digital keys. It is usually harder to read out, predict, or derive their responses than to obtain the values of digital keys stored in non-volatile memory. This fact has been exploited for various PUF-based security protocols, for example schemes for identification [1] and key exchange [2].

Oblivious Transfer. Oblivious transfer (OT) is a two-player cryptographic primitive which was originally introduced by [3] [4]. Several variants exist, which are reducible to each other [5] [31]. The version considered in this paper is a one-out-of-two oblivious transfer or $\binom{2}{1}$-OT [5]. This is a protocol with the following properties: At the beginning of the protocol, one party Alice (the "sender") holds two secret bits b_0 and b_1 as private input, and another party Bob (the "receiver") holds a secret choice bit c as private input. After execution of the protocol, the following conditions must be met: (i) Bob has learned the bit b_c, i.e. those of the two bits b_0 and b_1 that was selected by his choice bit c. (ii) Even an actively cheating Bob cannot derive any information about the other bit $b_{c\oplus 1}$ as long as Alice follows the protocol. (iii) Even an actively cheating Alice cannot learn c if Bob follows the protocol.

Since its introduction, a large class of cryptographic schemes has been realized on the basis of OT, including bit-commitment, zero-knowledge proofs, and general secure multi-party computation [6] [7] [8] [9] [10]. This makes OT a very versatile and universal primitive. The fact that OT can be realized within a certain cryptographic model is often seen as an indication of the model's large cryptographic potential. For these reasons, the feasibility of OT in the context of quantum cryptography [11] [12], within the Bounded Storage Model (BSM) [14] [15], or in noise-based cryptography [16] [17], has been well-investigated in earlier publications .

Our Contribution. In this extended abstract, we describe a protocol that implements oblivious transfer on the basis of two types of Physical Unclonable Functions: So-called Strong PUFs and SHIC PUFs (see Section 2). The protocol seems to indicate the large potential of these PUFs beyond the known schemes for identification [1] and key exchange [2].

The protocol can be executed between two players Alice and Bob under the following prerequisites: (i) Bob had previous access to the Strong PUF/SHIC PUF in a pre-setting phase. During this phase, he established a list of challenge-response-pairs (CRPs) of the PUF, which is unknown to Alice. (ii) At the time of protocol execution, the Strong PUF/SHIC PUF has been transfered to Alice. Only Alice has access to it and can measure CRPs of the PUF.

Since it is known from other publications that OT is a symmetric primitive [31], our technique allows OT in both directions under the above provisions, without re-transferring the PUF from Alice to Bob (see Sec. 3.3).

Organization of the Paper. In Section 2, we give some background on the two specific PUF types which are relevant for this paper (i.e., Strong PUFs and SHIC PUFs). We also briefly discuss their implementation. In Section 3 we describe and analyze our protocol for oblivious transfer. We conclude the paper in Section 4.

2 Background on PUFs

We now give some background on the two PUF types relevant for this paper. Since SHIC PUFs are a special form of Strong PUFs, we start with an explanation of the latter.

2.1 Strong PUFs

A Strong PUF [1] [18] is a (partly) disordered physical system S, which can be excited with a finite number of external stimuli or challenges C_i, upon which it reacts with corresponding responses R_{C_i} [2]. The pairs (C_i, R_{C_i}) are usually called the challenge-response pairs (CRPs) of the PUF. Three security relevant properties of a Strong PUF S are the following:

(i) Due to the disordered microstructure of S, it must be practically infeasible to fabricate a physical clone S' of S, which has the same challenge-response behavior as S. This restriction shall even hold for the original manufacturer of S.
(ii) Due to the large number of possible challenges that can be applied to S, it must be practically infeasible to exhaustively measure all CRPs of S within a limited time frame on the order of weeks, months, or even years.
(iii) Due to the complicated internal interactions of S, it must be practically infeasible to devise a computer program that correctly predicts the response of S to a randomly chosen, previously unknown challenge with high probability. This should hold even if many other challenge-response pairs of S are known.

Together, conditions (i) to (iii) imply that the responses R_{C_i} of S can be determined correctly (with high probability) only by someone who has got direct physical access to the single, unique PUF S. Implementation examples of Strong PUFs include complex optical scatterers [1] or integrated circuits whose outputs depend on their internal, individual runtimes delays [21] [22] [23]. Also analog cellular arrays have been proposed recently [24].

It has been realized relatively early, however, that machine learning techniques are a natural and powerful tool that can potentially challenge the above security condition (iii). Successful attacks on several Strong PUF candidates have indeed been reported in [20] [21] [25] [26]. To rule out a potential susceptibility to algorithmic modeling attacks, SHIC PUFs have been introduced.

2.2 SHIC PUFs

SHIC PUFs are pronounced as "chique PUFs" and have been suggested in [27] [28] [29]. The acronym "SHIC" stands for Super High Information Content. They are Strong PUF (i.e. they possess the above properties (i) to (iii)) and have the following additional features:

(iv) They contain an extraordinarily high amount of response-relevant random information and have a very high information density.

[1] Strong PUFs also have been referred to simply as PUFs [19], as Physical Random Functions [19] [21] [22] , or, almost equivalently, as Physical One-Way Functions [1].

[2] Please note that the terminology "R_{C_i}" slightly deviates from the standard terminology "R_i" for PUFs. The new terminology is introduced here in order to make the description of Protocol 2 less ambiguous.

(v) Their read-out speed (i.e. the frequency by which they produce responses) is limited to low values. This limitation should not be enforced by an artificially slow access module or the like, which could potentially be circumvented or cut off by suitable invasive means. Rather, it must be an inherent property of the PUF's design and its physical properties.

(vi) The CRPs of a SHIC PUF are mutually independent. The pairwise mutual information of any two responses of theirs is zero.

SHIC PUFs can be imagined as a huge read-only memory with a very high random information content and an *intrinsically slow read-out speed*. A challenge C_i to a SHIC PUF is the analogue to the address in a classical memory, and the corresponding response R_{C_i} is similar to the bit-value stored under that address. A possible realization with concrete numbers for information content, information density and read-out speed will be discussed in Section 2.3.

Strong PUFs vs. SHIC PUFs. As emphasized earlier, all SHIC PUFs are Strong PUFs, but they possess the further properties (iv) to (vi) above. SHIC PUFs thus have the advantage that their security does not depend on the computational power and the machine learning capabilities of the attacker. As all their CRPs are independent of each other, they withstand prediction even by attackers with unlimited computational power until a complete read-out has been accomplished. Their security only depends on the CRPs known to an adversary vs. the overall number of CRPs of the PUF.

2.3 Realization of SHIC PUFs

Even though this is not the main topic of this manuscript, we will briefly discuss the practical realization of the theoretical concept of a SHIC PUF. One potential candidate are ALILE-based Crossbar PUFs, which have been introduced in [27] [28] [29]. We will only provide a short overview of this approach; much further detail can be found in [27] [28] [29].

Generating Randomness by the ALILE Process. Any SHIC PUF must contain a very large random information content. There are many physical processes that generate large entropy in solid-state systems, but one example that can eventually lead to integrated electrical realizations of SHIC PUFs is a process known as ALuminum-Induded Layer Exchange (ALILE) [27] [28] [29]. It is a simple, crystallization-based method that employs only inexpensive starting materials (amorphous silicon and aluminum). It result in polycrystalline films with p-type conduction, which exhibit a very large level of disorder and randomness (see Fig. 1 a). By adjusting the process parameters, the size, number and density of the crystallites can be tuned as desired. The randomness causes individual electrical properties in different subregions of the surface.

Crossbar-based Read-Out. One method that was investigated in [27] [28] [29] is to read out the information from ALILE structures by so-called crossbar architectures. Slightly simplifying, a crossbar consists of two sets of parallel wires, which are attached to the top and to the bottom of the crystallized structure. The bottom set of wires is arranged in a 90° angle to the top set, as shown in Figure 1. If source and drain voltages are applied at exactly one top and one bottom wire, current flows through the polycrystalline film area at the virtual crossing of the two wires. $I(V)$ curves with a strongly rectifying behavior [29] are observed, which depend on the individual, random configuration in the polycrystalline substrate at the crossing. They can be converted into a few bits of individual information per crossing [27] [28].

Crossbar architectures are among the simplest functional nano devices and possess a very regular geometry. They can hence be fabricated with very small inter-wire distances, leading to high information densities. Concrete realization parameters we tried to make plausible by measurement on single diodes and by crossbar simulations in [28] are 10^5 top wires and 10^5 bottom wires, which leads to an information of around 10^{10} bits per cm^2. This assumes that the footprint of one crossing is 100 nm $\times$ 100 nm [28]. A single CRP of such a structure would have a length of around around $1 + 2 \cdot \log_2 10^5 \approx 35$ bits.

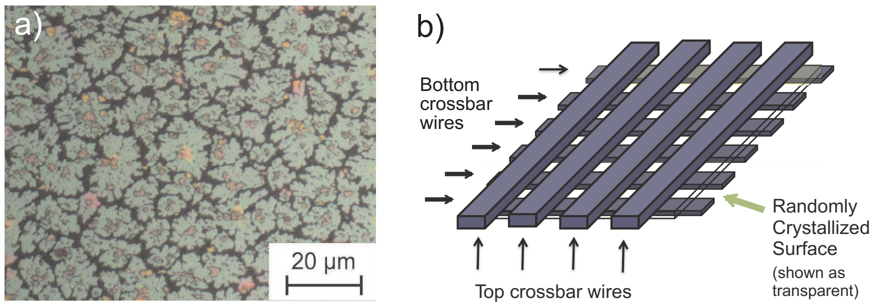


Fig. 1. a) A polycrystalline film resulting from the ALILE process, illustrating the high entropy and disorder in the structure. The green areas are silicon crystallites, possessing a random distribution and strongly irregular shape. b) The schematics of the crossbar read-out circuitry.

Inherently Slow Read-Out Speed. Up to now, we have mainly described a memory-like structure with a high information content and density. Also large arrays of SRAM cells or Butterfly PUFs could fulfill these criteria, albeit presumably at lower information densities. The perhaps most unusual characteristic of Crossbar PUFs is that they promise to guarantee an inherently slow read-out speed [28]. To achieve this property, the Crossbar PUF must be built in one large, monolithic block, not from separate blocks as modern semiconductor memories. The wires are intentionally designed to have only a low, limited current-carrying capacity. Simulations conducted in [28] showed that in such large blocks, depending on the fabrication parameters, several milliseconds must elapse before the sense current/voltage stabilizes. This leads to read-out speeds of around 100 bits/sec [28].

The two apparent strategies to accelerate read-out would be to increase the sense current/voltage, or to conduct a parallel read-out at several crossings. But both approaches lead to a higher current load in the monolithic crossbar, which is proportional to the achieved speed up. They therefore quickly overload and destroy the limited wires [28]. Removing the original wires of the crossbar, which very densely cover the whole crystallized system, and replacing them with a faster read-out mechanism seems practically infeasible without destroying the PUF's structure and current-voltage characteristics. This makes the PUF's original responses unreadable [28].

3 The Protocol

We now provide a protocol for $\binom{2}{1}$-OT on the basis of Strong PUFs, which is inspired by techniques originally presented in [14]. Since SHIC PUFs are a subclass of Strong PUFs, the protocol works for both PUF types interchangeably — using SHIC PUFs only causes some small advantages in the resulting security features (see section 3.3). As a subprotocol, we employ interactive hashing [13] [14].

3.1 Interactive Hashing

In a nutshell, interactive hashing [13] is a cryptographic two-player protocol, in which Alice has no input, and Bob's initial input is an m-bit string S. At the end of the protocol, Alice knows two m-bit strings U_1 and U_2, with the properties that (i) $U_b = S$ for some bit $b \in \{0, 1\}$, but Alice does not know the value of b, and that (ii) the other string $U_{b\oplus 1}$ is an essentially random bitstring of length m, which neither Alice nor Bob can determine alone. A protocol for interactive hashing can be constructed as follows.

Protocol 1: INTERACTIVE HASHING

Prerequisites:

1. Alice holds no input, Bob holds an m-bit string S as input.
2. Let G be the following class of 2-universal hash functions:

$$G = \{g(x) = a * x \mid a \text{ is an element of the set } \{0, 1\}^m\} ,$$

   where $*$ denotes the scalar product between the vectors a and x.

Protocol:

The protocol consists of $m - 1$ rounds. In the j-th round, for $j = 1, \ldots, m - 1$, Alice executes the following steps:

1. Alice chooses a function g_j uniformly at random from the set G. Let the m-ary binary vector a_j be the description of G. If a_j is linearly dependent on the $a_1, \ldots, a_{m-1}$, then Alice repeats step 1 until a_j is linearly independent.
2. Alice announces g_j to Bob.
3. Bob computes $b_j = g_j(S) = a_j * S$ and sends b_j to Alice.

At the end of the protocol, Alice knows $m - 1$ linear equations satisfied by S. Since the a_j's are linearly independent, there are exactly two different m-bit strings U_1 and U_2 that satisfy the system of equations set up by Bob. These solutions can be found by Alice via standard linear algebra. U_1 and U_2 have the property that exactly one of them is equal to S, but obviously Alice has no chance in telling which one it is. For further details see [13] [14].

3.2 Oblivious Transfer

Protocol 2: $\binom{2}{1}$-OBLIVIOUS TRANSFER BY STRONG PUFS

Prerequisites:

1. Bob holds a Strong PUF S. We assume without loss of generality that the responses R_C^S of S consist of a single bit. [3]
2. Alice and Bob have agreed on an encoding scheme $E(\cdot)$ with the following properties:
   (a) $E(\cdot)$ efficiently encodes finite tuples of PUF-challenges C_i of the form $T = (C_1, \ldots, C_k)$ as finite binary strings.
   (b) $E(\cdot)$ is reversed by a decoding scheme $D(\cdot)$, such that $E(D(T)) = T$ for all tuples T of the form $T = (C_1, \ldots, C_k)$ (with the C_i being challenges of S).
   (c) $D(\cdot)$ uniquely associates a tuple $T = D(x)$ with any finite binary string x.

   Similar encoding schemes can be found, for example, in [32] or [14].
3. Alice holds two bits b_0 and b_1, which are unknown to Bob.
4. Bob holds a choice bit c, which is unknown to Alice.

Protocol:

1. Bob chooses a tuple of challenges $T = (C_1, \ldots, C_n)$ uniformly at random, and determines the corresponding responses $R_{C_1}, \ldots, R_{C_n}$.
2. Bob sends or transfers the Strong PUF S to Alice.
3. Alice and Bob get engaged in an interactive hashing protocol, where Bob's input is $E(T)$.
4. The output of this interactive hashing protocol, which is both known to Alice and Bob, are two strings U_0 and U_1. One of these strings U_0, U_1 is equal to $E(T)$. Let us call the index of that string i_0, i.e. $U_{i_0} = E(T)$.

   Note: Bob knows i_0, since he knows both U_0, U_1 and $E(T)$.

5. Bob sets the bit $c' = i_0 \oplus c$, and sends c' to Alice.
6. Alice determines by measurement on the PUF S the values

$$R_{Z_1}, \ldots, R_{Z_n},$$

   where the Z_i are the elements of the tuple $D(U_{c'})$ (which, by the properties of $D(\cdot)$, are all challenges of S). Furthermore, she determines by measurement on S the values

$$R_{Z_1{}'}, \ldots, R_{Z_n{}'},$$

   where the $Z_i{}'$ are the elements of the set $D(U_{c' \oplus 1})$.

[3] If a response consists of multiple bits $b_1 \cdots b_k$, we can, for example, take the XOR of all these bits, or employ fuzzy extractors.

Note: At this point of the protocol, Alice has chosen two sets of PUF-responses $R_{Z_1}, \ldots, R_{Z_n}$ and $R_{Z_1}{}', \ldots, R_{Z_n}{}'$. Bob knows exactly one of these sets, namely the one that is equal to $R_{C_1}, \ldots, R_{C_n}$. The other set is unknown to Bob. Furthermore, Alice does not know which of the two sets of responses is known to Bob.

7. Alice forms the two strings s_0 and s_1 according to the following rules:

$$s_0 = b_0 + R_{Z_1} + \ldots + R_{Z_n} \mod 2,$$

and

$$s_1 = b_1 + R_{Z_1}{}' + \ldots + R_{Z_n}{}' \mod 2.$$

8. Alice sends s_0 and s_1 to Bob.
9. Bob obtains the bit b_c he selected through his choice bit c as

$$b_c = s_c + R_{C_1} + \ldots + R_{C_n} \mod 2.$$

3.3 Discussion

Security. The security of the protocol depends on the fact that Bob does not know both sets $R_{Z_1}, \ldots, R_{Z_n}$ and $R_{Z_1}{}', \ldots, R_{Z_n}{}'$ in step 7. If he did, then he could learn both bits b_0 and b_1. This is where property (iii) (see page 3) of Strong PUFs and SHIC PUFs becomes relevant. Due to this property, Bob cannot know all CRPs of the Strong PUF/SHIC PUF, but only a fraction γ with $0 < \gamma < 1$. Since one of the sets $R_{Z_1}, \ldots, R_{Z_n}$ and $R_{Z_1}{}', \ldots, R_{Z_n}{}'$ is chosen at random in the interactive hashing protocol, the probability that Bob knows the corresponding CRPs is γ^n, i.e. it is exponentially low in the security parameter n of the protocol. The fact that Alice does not learn Bob's choice bit c stems from the security properties of the interactive hashing protocol, which prevents that Alice learns which of the two strings U_1 or U_2 is equal to Bob's private input S [13] [14].

The security difference in using Strong PUFs and SHIC PUFs in Protocol 2 is that by its definition and property (vi), a secure SHIC PUF would fulfill the essential requirement (iii) (see page 3) independent of the computational power of the adversary. Secure SHIC PUFs hence could guarantee the protocol's security also against cheating parties with unlimited computational potential.

Practicality. The communication and storage requirements are mild: Bob must store only n CRPs, and the protocol has around m rounds for the interactive hashing. The latter can be reduced to a constant the techniques described in [15].

In order to cope with potential noise in the PUF responses, presumably standard PUF error correction such as helper data (see [2] [27] and references therein) could be used. In that case, a few steps of the protocol should be adjusted. Firstly, Bob measures and stores noisy data R_{C_i} in Step 1. Alice likewise obtains noisy responses R_{Z_i} and $R_{Z_i}{}'$ in Step 6 of the protocol, and extracts helper data W_{Z_i} and $W_{Z_i}{}'$, together with secrets S_{Z_i} and $S_{Z_i}{}'$. In Step 7, Alice uses the secrets S_{Z_i} and $S_{Z_i}{}'$ (instead of the values R_{Z_i} and $R_{Z_i}{}'$) to "encrypt" the bits b_0 and b_1. In Step 8, she transmits the corresponding helper data W_{Z_i} and $W_{Z_i}{}'$ together with the strings s_0 and s_1. Of these two sets of helper data, Bob uses the one that matches his data set R_{C_i}. He derives identical secrets as Alice from the R_{C_i}, and uncovers the bit b_c from $s_{i_0 \oplus c}$.

Symmetry. Oblivious transfer is known to be a symmetric primitive [31]: Given an OT protocol where Alice is the sender and Bob is the receiver, one can construct the "reverse" OT protocol where Alice acts as receiver and Bob as sender. The construction of [31] is generic, and independent of the concrete implementation of the OT.

Therefore, Protocol 2 can also be used to implement OT in the other direction, i.e. from Bob to Alice, without re-transferring the PUF from Alice to Bob. This is an important practicality asset: In many applications, the physical transfer of the PUF in one direction is executed naturally (e.g. in a hardware shipped from a manufacturer to a customer, or on a bank card carried to an automated teller machine (ATM) by a customer). Once accomplished, this allows oblivious transfer in both directions and secure two-party computations, e.g. between the manufacturer and the hardware.

4 Summary

We discussed a protocol for oblivious transfer on the basis of Strong PUFs and SHIC PUFs. It allows OT and secure two-party computation between two players, provided that (i) Player A had previous access to the PUF, and (ii) only Player B holds physical possession of the PUF at the time of the protocol execution. These circumstances occur frequently in practice, for example between a central authority on the one hand and mobile hardware systems, decentral terminals, or security tokens (including bank cards, ID cards, access cards, and the like) on the other hand. The protocol does not use any computational assumptions other than the security of the PUF.

References

1. R. Pappu, B. Recht, J. Taylor, N. Gershenfeld: *Physical One-Way Functions*, Science, vol. 297, pp. 2026-2030, 20 September 2002.
2. 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.
3. M. O. Rabin: *How to exchange secrets by oblivious transfer.* Technical Report TR-81, Harvard University, 1981.
4. S. Even, O. Goldreich, A. Lempel: *A randomized protocol for signing contracts.* In: Proc. CRYPTO 82 (R. L. Rivest, A. Sherman, S. Chaum, Eds.), pp. 205-210, Plenum Press, 1983.
5. C. Crepeau: *Equivalence between two flavors of oblivious transfer.* CRYPTO '87 (C. Pomerance, Ed.), LNCS Vol. 293, pp. 350–354, Springer, 1988.
6. A. C.-C. Yao: *How to generate and exchange secrets.* Proc. of the 27th IEEE Symposium on the Foundations of Computer Science (FOCS), pp. 162–167, 1986.
7. O. Goldreich, S. Micali, A. Widgerson: *How to play any mental game, or a completeness theorem for protocols with honest majority.* Proc. of the 19th Annual Symposium on the Theory of Computing (STOC), pp. 218–229, 1987.

8. O. Goldreich, R. Vainish: *How to solve any protocol problem – an efficiency improvement.* CRYPTO 87 (C. Promenance, Ed.), LNCS Vol. 293, pp. 73-86, Springer 1988.
9. J. Kilian: *Founding cryptography on oblivious transfer*. Proceedings, 20th Annual ACM Symposium on the Theory of Computation (STOC), 1988.
10. C. Crepeau, J. van de Graaf, A. Tapp: *Committed oblivious transfer and private multi-party computations*. CRYPTO 95, LNCS Vol. 963, pp. 110-123, Springer 1995.
11. G.P. He, Z.D. Wang: *Oblivious transfer using quantum entanglement*. Physical Review A 2006, VOL 73; NUMB 1; PART A, pages 012331.
12. S. Wehner, C. Schaffner, B.M. Terhal, *Cryptography from noisy storage*. Phys Rev Lett. 2008 Jun 6;100(22):220502.
13. M. Naor, R. Ostrovsky, R. Venkatesan, M. Yung: *Perfect zero-knowledge arguments for NP using any one-way function*. Journal of Cryptology 11 (1998), no. 2, 87–108.
14. C. Cachin, C. Crepeau, J. Marcil: *Oblivious transfer with a memory-bounded receiver.* Proceeding of the 39th Annual Symposium on Foundations of Computer Science, 1998.
15. Y.Z. Ding, D. Harnik, A. Rosen, R. Shaltiel: *Constant-Round Oblivious Transfer in the Bounded Storage Model*. Journal of Cryptology, 2007.
16. C. Crepeau: *Efficient cryptographic protocols based on noisy channels*. EUROCRYPT 97 (Walter Fumy, Ed.), LNCS Vol. 1233, pp. 306–317, Springer, 1997.
17. Jürg Wullschleger: *Oblivious Transfer from Weak Noisy Channels.* TCC 2009: 332-349
18. Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, Pim Tuyls: *FPGA Intrinsic PUFs and Their Use for IP Protection*. CHES 2007: 63-80
19. Blaise Gassend, *Physical Random Functions*, MSc Thesis, MIT, 2003.
20. Daihyun Lim: *Extracting Secret Keys from Integrated Circuits.* MSc Thesis, MIT, 2004.
21. 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.
22. J.-W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas. *A technique to build a secret key in integrated circuits with identification and authentication applications.* In Proceedings of the IEEE VLSI Circuits Symposium, June 2004.
23. M. Majzoobi, F. Koushanfar, M. Potkonjak: *Lightweight Secure PUFs.* IC-CAD 2008.
24. 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.
25. M. Majzoobi, F. Koushanfar, M. Potkonjak: *Testing Techniques for Hardware Security.* IEEE International Test Conference, 2008.
26. U. Rührmair, F. Sehnke, J. Soelter, S. Devadas, J. Schmidhuber: *Modeling Attacks on Physical Unclonable Functions.* Submitted, 2010.
27. U. Rührmair, C. Jaeger, C. Hilgers, M. Algasinger, G. Csaba, M. Stutzmann: *Security Applications of Diodes with Random Current-Voltage Characteristics.* Financial Cryptography and Data Security, 2010.
28. U. Rührmair, C. Jaeger, M. Bator, M. Stutzmann, P. Lugli, and G. Csaba: *Applications of High-Capacity Crossbar Memories in Cryptography.* To appear in IEEE Transactions on Nanotechnology, 2010.
29. C. Jaeger, M. Algasinger, U. Rührmair, G. Csaba, M. Stutzmann: *Random pn-junctions for physical cryptography.* To appear in Applied Physics Letters, 2010.
30. G. E. Suh, S. Devadas: *Physical Unclonable Functions for Device Authentication and Secret Key Generation.* DAC 2007: 9-14
31. S. Wolf, J. Wullschleger: *Oblivious Transfer Is Symmetric.* EUROCRYPT 2006: 222-232
32. T. M. Cover: *Enumerative Source Encoding*. IEEE Transactions on Information Theory 19 (1973), No. 1, pp. 73 – 77.

Chapter 4

Practical Security Analysis of PUF-based Two-Player Protocols

In classical cryptography, security proofs are often led in an asymptotic framework, where the differentiation between exponential and polynomial complexities plays a crucial role. This formalism has been applied successfully for several decades in classical cryptography. This makes it tempting to directly transfer this formalism to PUFs.

On closer inspection, however, such a step turns out to be problematic. The most obvious concern is that PUFs are finite physical systems, which cannot be scaled indefinitely due to size, cost, and stability constraints. Take Pappu et al.'s well-known optical PUF as an example: If its size of $1\text{cm}\times 1\text{cm}$ was scaled by a factor of 10 on each side, it already would become unusable for most standard applications, for example for bank cards and smart cards. Furthermore, if its thickness was scaled too much, not enough light would leave the structure any longer, making it difficult or even infeasible to collect stable responses. Similar considerations apply to other, electrical PUFs: Scaling the size too strongly is infeasible for several practical reasons, and may also have a comparably small additional effect on the security level. Consequentially, the constants in the security guarantees suddenly matter when we consider PUFs, at least much more strongly than in classical, complexity-based mathematical cryptography.

This chapter illustrates that these observations are not just of theoretical concern, but that a pure focus on asymptotics can lead to PUF schemes that are provable secure in theory, but demonstrably insecure in practice. More precisely, we show that protocols for oblivious transfer (OT) and bit commitment (BC) suggested by Brzuska, Fischlin, Schröder and Katzenbeisser at CRYPTO 2011 [6] allow quadratic attacks. These attacks are obviously not relevant in a typical asymptotic, polynomial-vs.-exponential framework. But they do make the protocol insecure in practice when it is used in connection with optical PUFs, as suggested explicitly at CRYPTO 2011 [6]. Our attacks hence constitutes a real security concern that is not monitored by current, asymptotic PUF formalisms.

We subsequently observe that our earlier OT-protocol from Chapter 3, which utilized interactive hashing as a substep, can be used with a higher security level. Along these lines, we present a simplified and shortened version of the OT-protocol in this chapter.

The employed paper in this chaper is:

- U. Rührmair, M. van Dijk: *Practical Security Analysis of PUF-based Two-Player Protocols*. Cryptographic Hardware and Embedded Systems (CHES 2012), pp. 251-267, Lecture Notes in Computer Science Springer, Vol. 7428, 2012.

The work has been selected as one of the best papers of CHES 2012, and was invited for a journal version at the Journal of Cryptographic Engineering. According to Google scholar, it has been cited 23 times to this date [26].

The candidate would again like to express his gratitude to have one of the founding fathers of PUFs as his co-author in this paper.

Practical Security Analysis of PUF-Based Two-Player Protocols

Ulrich Rührmair[1] and Marten van Dijk[2]

[1] Technische Universität München, 80333 München, Germany
ruehrmair@in.tum.de
[2] RSA Laboratories, Cambridge, MA, USA
marten.vandijk@rsa.com

Abstract. In recent years, PUF-based schemes have not only been suggested for the basic tasks of tamper sensitive key storage or the identification of hardware systems, but also for more complex protocols like oblivious transfer (OT) or bit commitment (BC), both of which possess broad and diverse applications. In this paper, we continue this line of research. We first present an attack on two recent OT- and BC-protocols which have been introduced at CRYPTO 2011 by Brzuska et al. [1,2]. The attack quadratically reduces the number of CRPs which malicious players must read out in order to cheat, and fully operates within the original communication model of [1,2]. In practice, this leads to insecure protocols when electrical PUFs with a medium challenge-length are used (e.g., 64 bits), or whenever optical PUFs are employed. These two PUF types are currently among the most popular designs. Secondly, we discuss countermeasures against the attack, and show that interactive hashing is suited to enhance the security of PUF-based OT and BC, albeit at the price of an increased round complexity.

Keywords: Physical Unclonable Functions (PUFs), Cryptographic Protocols, Oblivious Transfer, Bit Commitment, Security Analysis, Interactive Hashing.

1 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 physical 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 [22], 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)* was 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 $r = f_P(c)$. The 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, and 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 [21,9], key exchange [21], and various forms of (tamper sensitive) key storage and applications thereof, such as intellectual property protection or read-proof memory [11,14,29].

In recent years, also the use of PUFs in more advanced cryptographic protocols together with formal security proofs has been investigated. In these protocols, usually PUFs with a large challenge set and with a freely accessible challenge-response interface are employed.[1] The PUF is used similar to a "physical random oracle", which is transferred 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 PUF at earlier points in time. In 2010, Rührmair [23] showed that oblivious transfer (OT) can be realized between two parties by physically transferring a PUF in this setting. He observed that via the classical reductions of Kilian [13], this implies PUF-based bit commitment and PUF-based secure multi-party computations. In the same year, the first formal security proof for a PUF-protocol was provided by Rührmair, Busch and Katzenbeisser [24]. They presented definitions and a reductionist security proof for PUF-based identification protocols. At CRYPTO 2011 Brzuska et al. [1] adapted Canetti's universal composition (UC) framework [3] to include PUFs. They gave PUF-based protocols for oblivious transfer (OT), bit commitment (BC) and key exchange (KE) and proved them to be secure in their framework.

The investigation of advanced cryptographic settings for PUF makes sense even from the perspective of a pure practitioner: Firstly, it clarifies the potential of PUFs in theory, a necessary prerequisite before this potential can be unleashed in commercial applications without risking security failures. Secondly, BC and OT protocols are

[1] This type of PUF sometimes has been termed *Physical Random Function* [9] or *Strong PUF* [11,26,25,24] in the literature. We emphasize that the Weak/Strong PUF terminology introduced by Guajardo et al. [11] is not to be understood in a judgemental or pejorative manner.

extremely versatile cryptographic primitives, which allow the implementation of such diverse tasks as zero-knowledge identification, the enforcement of semi-honest behavior in cryptographic protocols, secure multi-party computation (including online auctions or electronic voting), or key exchange. If these tasks shall be realized securely in practice by PUFs, a theoretical investigation of the underlying primitives — in this case BC and OT — is required first.

In this paper, we continue this line of research, and revisit the use of PUFs in OT- and BC-protocols. Particular emphasis is placed on the achievable *practical security* if well-established PUFs (like electrical PUFs with 64-bit challenge lengths or optical PUFs) are used in the protocols. We start by observing an attack on the OT- and BC-protocols of Brzuska et al. [1,2] which quadratically reduces the number of responses that a malicious player must read out in order to cheat. It works fully in the original communication model of Brzuska et al. and makes no additional assumptions. As we show, the attack makes the protocols insecure in practice if electrical PUFs with medium bitlengths around 64 bits are used, and generally if optical PUFs are employed. This has a special relevance since the use of optical PUFs for their protocols had been explicitly proposed by Brzuska et al. (see Section 8 of [2]). Secondly, we investigate countermeasures against our attack, and show that interactive hashing can be used to enhance the security of PUF-based OT and BC protocols.

Our work continues the recent trend of a formalization of PUFs, including protocol analyses, more detailed investigations of non-trivial communication settings, and formal security proofs. This trend will eventually lay the foundations for future PUF research, and seems indispensible for a healthy long-term development of the field. It also combines protocol design and practical security analyses in a novel manner.

Organization of this Paper. In Section 2 we present the protocols of Brzuska et al. in order to achieve a self-contained treatment. Section 3 gives our quadratic attack. Section 4 discusses its practical effect. Section 5 discusses countermeasures. We conclude the paper in Section 6.

2 The Protocols of Brzuska et al.

Our aim in this paper is to present a quadratic attack on two recent PUF-protocols for OT and BC by Brzuska et al. [1,2] and to discuss its practical relevance. In order to achieve a self-contained treatment, we will now present these two protocols. To keep our exposition simple, we will not use the full UC-notation of [1], and will present the schemes mostly without error correction mechanisms, since the latter play no role in the context of our attack.

The protocols use two communication channels between the communication partners: A binary channel, over which all digital communication is handled. It is assumed that this channel is non-confidential, but authenticated. And secondly an insecure physical channel, over which the PUF is sent. It is assumed that adversaries can measure adaptively selected CRPs of the PUF while it is in transition over this channel.

2.1 Oblivious Transfer

The OT protocol of [1] implements one-out-of-two string oblivious transfer. It is assumed that in each subsession the sender P_i initially holds two (fresh) bitstrings $s_0, s_1 \in \{0,1\}^\lambda$, and that the receiver P_j holds a (fresh) choice bit b.

Brzuska et al. generally assume in their treatment that after error correction and the application of fuzzy extractors, a PUF can be modeled as a function $\mathsf{PUF} : \{0,1\}^\lambda \rightarrow \{0,1\}^{rg(\lambda)}$. We use this model throughout this paper, too. In the subsequent protocol of Brzuska et al., it is furthermore assumed that $rg(\lambda) = \lambda$, i.e., that the PUF implements a function $\mathsf{PUF} : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$ (compare [1,2]).

Protocol 1: PUF-BASED OBLIVIOUS TRANSFER ([1], SLIGHTLY SIMPLIFIED DESCRIPTION)

External Parameters: The protocol has a number of external parameters, including the security parameter λ, the session identifier `sid`, a number N that specifies how many subsessions are allowed, and a pre-specified PUF-family $\mathcal{P}$, from which all PUFs which are used in the protocol must be drawn.

Initialization Phase: Execute once with fixed session identifier `sid`:

1. The receiver holds a PUF which has been drawn from the family $\mathcal{P}$.
2. The receiver measures l randomly chosen CRPs $c_1, r_1, \ldots, c_l, r_l$ from the PUF, and puts them in a list $\mathcal{L} := (c_1, r_1, \ldots, c_l, r_l)$.
3. The receiver sends the PUF to the sender.

Subsession Phase: Repeat at most N times with fresh subsession identifier `ssid`:

1. The sender's input are two strings $s_0, s_1 \in \{0,1\}^\lambda$, and the receiver's input is a bit $b \in \{0,1\}$.
2. The receiver chooses a CRP (c, r) from the list $\mathcal{L}$ at random.
3. The sender chooses two random bitstrings $x_0, x_1 \in \{0,1\}^\lambda$ and sends x_0, x_1 to the receiver.
4. The receiver returns the value $v := c \oplus x_b$ to the sender.
5. The sender measures the responses r_0 and r_1 of the PUF that correspond to the challenges $c_0 := v \oplus x_0$ and $c_1 := v \oplus x_1$.
6. The sender sets the values $S_0 := s_0 \oplus r_0$ and $S_1 := s_1 \oplus r_1$, and sends S_0, S_1 to the receiver.
7. The receiver recovers the string s_b that depends on his choice bit b as $s_b = S_b \oplus r$. He erases the pair (c, r) from the list $\mathcal{L}$.

Comments. The protocol implicitly assumes that the sender and receiver can interrogate the PUF whenever they have access to it, i.e., that the PUF's challenge-response interface is publicly accessible and not protected. This implies that the employed PUF must possess a large number of CRPs. Using a PUF with just a few challenges does not make sense: The receiver could then create a full look-up table for all CRPs of such a PUF before sending it away in Step 3 of the Initialization Phase. This would subsequently allow him to recover both strings s_0 and s_1 in Step 6 of the protocol subsession, as he could obtain r_0 and r_1 from his look-up table. Similar observations hold for the upcoming protocol 2. Indeed, all protocols discussed in this paper require PUFs with a large number of challenges and publicly accessible challenge-response interfaces. These PUFs have sometimes been referred to as *Physical Random Functions* or also as *Strong PUFs* in the literature [11,26,25].

Furthermore, please note that no physical transfer of the PUF is envisaged during the subsessions of the protocol. According to the model of Brzuska et al., an adversary only has access to it during the initialization phase, but not between the subsessions. This protocol use has some similarities with a stand-alone usage of the PUF, in which exactly one PUF-transfer occurs between the parties.

2.2 Bit Commitment

The second protocol of [1] implements PUF-based Bit Commitment (BC) by a generic reduction to PUF-based OT. The BC-sender initially holds a bit b. When the OT-Protocol is called as a subprotocol, the roles of the sender and receiver are reversed: The BC-sender acts as the OT-receiver, and the BC-receiver as the OT-sender. The details are as follows.

Protocol 2: PUF-BASED BIT COMMITMENT VIA PUF-BASED OBLIVIOUS TRANSFER ([1], SLIGHTLY SIMPLIFIED DESCRIPTION)

Commit Phase:

1. The BC-sender and the BC-receiver jointly run an OT-protocol (for example Protocol 1).
   (a) In this OT-protocol, the BC-sender acts as OT-receiver and uses his bit b as the choice bit of the OT-protocol.
   (b) The BC-receiver acts as OT-sender. He chooses two strings $s_0, s_1 \in \{0,1\}^\lambda$ at random, and uses them as his input s_0, s_1 to the OT-protocol.
2. When the OT-protocol is completed, The BC-sender has learned the string $v := s_b$. This closes the commit phase.

Reveal Phase:

1. In order to reveal bit b, the BC-sender sends the string (b, v) (with $v = s_b$) to the BC-receiver.

Comments. The security of the BC-protocol is inherited from the underlying OT-protocol. Once this protocol is broken, also the security of the BC-protocol is lost. This will be relevant in the upcoming sections.

3 A Quadratic Attack on Protocols 1 and 2

We will now discuss a cheating strategy in Protocols 1 and 2. Compared to an attacker who exhaustively queries the PUF for all of its m possible challenges, we describe an attack on Protocols 1 and 2 which reduces this number to $\sqrt{m}$. As we will argue later in Section 4, this has a particularly strong effect on the protocol's security if an optical PUF is used (as has been explicitly suggested by [2]), or if electrical PUFs with medium challenge lengths of 64 bits are used.

Our attack rests on the following lemma.

Lemma 3. *Consider the vector space $(\{0,1\}^\lambda, \oplus)$, $\lambda \geq 2$, with basis $\mathcal{B} = \{a_1, \ldots, a_{\lfloor \lambda/2 \rfloor}, b_1, \ldots, b_{\lceil \lambda/2 \rceil}\}$. Let A be equal to the linear subspace generated by the vectors in $\mathcal{B}_A = \{a_1, \ldots, a_{\lfloor \lambda/2 \rfloor}\}$, and let B be the linear subspace generated by the vectors in $\mathcal{B}_B = \{b_1, \ldots, b_{\lceil \lambda/2 \rceil}\}$. Define $S := A \cup B$. Then it holds that:*

(i) *Any vector $z \in \{0,1\}^\lambda$ can be expressed as $z = a \oplus b$ with $a, b \in S$, and this expression (i.e., the vectors a and b) can be found efficiently (i.e., in at most poly(λ) steps).*
(ii) *For all distinct vectors $x_0, x_1, v \in \{0,1\}^\lambda$ there is an equal number of combinations of linear subspaces A and B as defined above for which $x_0 \oplus v \in A$ and $x_1 \oplus v \in B$.*
(iii) *S has cardinality $|S| \leq 2 \cdot 2^{\lceil \lambda/2 \rceil}$.*

Proof. (i) Notice that any vector $z \in \{0,1\}^\lambda$ can be expressed as a linear combination of all basis vectors: $z = \sum u_i a_i + \sum v_j b_j$, i.e., $z = a \oplus b$ with $a \in A$ and $b \in B$. This expression is found efficiently by using Gaussian elimination.

(ii) Without loss of generality, since x_0, x_1 and v are distinct vectors, we may choose $a_1 = x_0 \oplus v \neq 0$ and $b_1 = x_1 \oplus v \neq 0$. The number of combinations of linear subspaces A and B is independent of the choice of a_1 and b_1. (Notice that if $x_0 \neq x_1$ but $v = x_0$, then the number of combinations is twice as large.)

(iii) The bound follows from the construction of S and the cardinalities of A and B, which are $|A| = 2^{\lfloor \lambda/2 \rfloor}$ and $|B| = 2^{\lceil \lambda/2 \rceil}$. □

An Example. Let us give an example in order to illustrate the principle of Lemma 3. Consider the vector space $(\{0,1\}^\lambda, \oplus)$ for an even λ, and choose as subbases $\mathcal{B}_{A_0} = \{e_1, \ldots, e_{\lambda/2}\}$ and $\mathcal{B}_{B_0} = \{e_{\lambda/2+1}, \ldots, e_\lambda\}$, where e_i is the unit vector of length λ that has a one in position i and zeros in all other positions. Then the basis $\mathcal{B}_{A_0}$ spans the subspace A_0 that contains all vectors of length λ whose second half is all zero, and $\mathcal{B}_{B_0}$ spans the subspace B_0 that comprises all vectors of length λ whose first half is all zero. It then follows immediately that every vector $z \in \{0,1\}^\lambda$ can be expressed as $z = a \oplus b$ with $a \in A_0$ and $b \in B_0$, or, saying this differently, with $a, b \in S$ and $S := A_0 \cup B_0$. It is also immediate that S has cardinality $|S| \leq 2 \cdot 2^{\lambda/2}$.

Relevance for PUFs. The lemma translates into a PUF context as follows. Suppose that a malicious and an honest player play the following game. The malicious player gets access to a PUF with challenge length λ in an initialization period, in which he can query CRPs of his choice from the PUF. After that, the PUF is taken away from him. Then, the honest player chooses a vector $z \in \{0,1\}^\lambda$ and sends it to the malicious player. The malicious player wins the game if he can present the correct PUF-responses r_0 and r_1 to two arbitrary challenges c_0 and c_1 which have the property that $c_0 \oplus c_1 = z$. Our lemma shows that in order to win the game with certainty, the malicious player does not need to read out the entire CRP space of the PUF in the initialization phase; he merely needs to know the responses to all challenges in the set S of Lemma 3, which has a quadratically reduced size compared to the entire CRP space. This observation is at the heart of the attack described below.

In order to make the attack hard to detect for the honest player, it is necessary that the attacker chooses random subspaces A and B, and does not use the above trivial choices A_0 and B_0 all the time. This fact motivates the random choice of A and B in Lemma 3. The further details are as follows.

The Attack. As in [1,2], we assume that the PUF has got a challenge set of $\{0,1\}^\lambda$. Given Lemma 3, the OT-receiver (who initially holds the PUF) can achieve a quadratic advantage in Protocol 1 as described below.

First, he chooses uniformly random linear subspaces A and B, and constructs the set S, as described in Lemma 3. While he holds possession of the PUF before the start of the protocol, he reads out the responses to all challenges in S. Since $|S| \leq 2 \cdot 2^{\lceil \lambda/2 \rceil}$, this is a quadratic improvement over reading out all responses of the PUF.

Next, he starts the protocol as normal. When he receives the two values x_0 and x_1 in Step 3 of the protocol, he computes two challenges c_0^* and c_1^* both in set S such that

$$x_0 \oplus x_1 = c_0^* \oplus c_1^*.$$

According to Lemma 3(i), this can be done efficiently (i.e., in $poly(\lambda)$ operations). Notice that, since the receiver knows all the responses corresponding to challenges in S, he in particular knows the two responses r_0^* and r_1^* that correspond to the challenges c_0^* and c_1^*.

Next, the receiver deviates from the protocol and sends the value $v := c_0^* \oplus x_0$ in Step 4. For this choice of v, the two challenges c_0 and c_1 that the sender uses in Step 5 satisfy

$$c_0 := c_0^* \oplus x_0 \oplus x_0 = c_0^*$$

and

$$c_1 := c_0^* \oplus x_0 \oplus x_1 = c_0^* \oplus c_0^* \oplus c_1^* = c_1^*.$$

By Lemma 3(ii), Alice cannot distinguish the received value v in Step 4 from any random vector v. In other words, Alice cannot distinguish Bob's malicious behavior (i.e., fabricating a special v with suitable properties) from honest behavior. As a consequence, Alice continues with Step 6 and transmits $S_0 = s_0 \oplus r_0^*$ and $S_1 = s_1 \oplus r_1^*$. Since Bob knows both r_0^* and r_1^*, he can recover both s_0 and s_1. This breaks the security of the protocol.

Please note the presented attack is simple and effective: It fully works within the original communication model of Brzuska et al. [1,2]. Furthermore, it does not require laborious computations of many days on the side of the attacker (as certain modeling attacks on PUFs do [25]). Finally, due to the special construction we proposed, the honest players will not notice the special choice of the value v, as the latter shows no difference from a randomly chosen value.

Effect on Bit Commitment (Protocol 2). Due to the reductionist construction of Protocol 2, our attack on the oblivious transfer scheme of Protocol 1 directly carries over to the bit commitment scheme of Protocol 2 if Protocol 1 is used in it as a subprotocol. By using the attack, a malicious sender can open the commitment in both ways by reading out only $2 \cdot 2^{\lceil \lambda/2 \rceil}$ responses (instead of all 2^{λ} responses) of the PUF. On the other hand it can be observed easily that the hiding property of the BC-Protocol 2 is unconditional, and is not affected by our attack.

4 Practical Consequences of the Attack

What are the practical consequences of our quadratic attack, and how relevant is it in real-world applications? The situation can perhaps be illustrated via a comparison to classical cryptography. What effect would a quadratic attack have on schemes like RSA, DES and SHA-1? To start with RSA, the effect of a quadratic attack here is rather mild: The length of the modulus must be doubled. This will lead to longer computation times, but restore security without further ado. In the case of single-round DES, however, a quadratic attack would destroy its security, and the same holds for SHA-1. The actual effect of our attack on PUF-based OT and BC has some similarities with DES or SHA-1: PUFs are finite objects, which cannot be scaled in size indefinitely due to area requirements, arising costs, and stability problems. This will also become apparent in our subsequent discussion.

4.1 Electrical Integrated PUFs

We start our discussion by electrical integrated PUFs, and take the well-known Arbiter PUF as an example. It has been discussed in theory and realized in silicon mainly for challenge lengths of 64 bits up to this date [9,10,15,28]. Our attack on such a 64-bit implementation requires the read-out of $2 \cdot 2^{32} = 8.58 \cdot 10^{9}$ CRPs by the receiver. This read-out can be executed before the protocol (i.e., not during the protocol), and will hence not be noticed by the sender. Assuming a MHz CRP read-out rate [15] of the Arbiter PUF, the read-out takes $8.58 \cdot 10^{3}\,\mathrm{sec}$, or less than $144\,\mathrm{min}$.

Please note that the attack is independent of the cryptographic hardness of the PUF, such as its resilience against machine learning attacks. For example, a 64-bit, 8-XOR-Arbiter PUF (i.e., an Arbiter PUF with eight parallel standard 64-bit Arbiter PUFs whose single responses are XORed at the end of the structure) is considered secure in practice against all currently known machine learning techniques [25]. Nevertheless, this type of PUF would still allow the above attack in $144\,\mathrm{min}$.

Our attacks therefore enforce the use of PUFs with a challenge bitlength of 128 bits or more in Protocols 1 and 2. Since much research currently focuses on 64-bit implementations of electrical PUFs, publication and dissemination of the attack seems important to avoid their use in Protocols 1 and 2. Another aspect of our attack is that it motivates the search for OT- and BC-protocols that are immune, and which can safely be used with 64-bit implementations. The reason is that the usage of 128-bit PUFs doubles the area consumption of the PUF and negatively affects costs.

4.2 Optical PUFs

Let us now discuss the practical effect of our attack on the optical PUF introduced by Pappu [20] and Pappu et al. [21]. The authors use a cuboid-shaped plastic token of size 1 cm $\times$ 1 cm $\times$ 2.5 mm, in which thousands of light scattering small spheres are distributed randomly. They analyze the number of applicable, decorrelated challenge-response pairs in their set-up, arriving at a figure of $2.37 \cdot 10^{10}$ [21]. Brzuska et al. assume that these challenges are encoded in a set of the form $\{0,1\}^{\lambda}$, in which case $\lambda = \lceil \log_2 2.37 \cdot 10^{10} \rceil = 35$. If this number of 2^{35} is reduced quadratically by virtue of Lemma 3, we obtain on the order of $2 \cdot 2^{18} = 5.2 \cdot 10^5$ CRPs that must be read out by an adversary in order to cheat. It is clear that even dedicated measurement set-ups for optical PUFs cannot realize the MHz rates of the electrical example in the last section. But even assuming mild read-out rates of 10 CRPs or 100 CRPs per second, we still arrive at small read-out times of $5.2 \cdot 10^4\,\mathrm{sec}$ or $5.2 \cdot 10^3\,\mathrm{sec}$, respectively. This is between 14.4 hours (for 10 CRPs per second) or 87 minutes (for 100 CRPs per second). If a malicious receiver holds the PUF for such a time frame before the protocol starts (which is impossible to control or prevent for the honest players), he can break the protocol's security.

Can the situation be cleared by simply scaling the optical PUF to larger sizes? Unfortunately, also an asymptotic analysis of the situation shows the same picture. All variable parameters of the optical PUF [21,20,16] are the x-y-coordinate of the incident laser beam and the spatial angle Θ under which the laser hits the token. This leads to a merely cubic complexity in the three-dimensional diameter d of the cuboid scattering token. [2] Given our attack, this implies that the adversary must only read out $O(d^{\,1.5})$ challenges in order to cheat in Protocols 1 and 2. If only the independent challenges are considered, the picture is yet more drastic: As shown in [31], the PUF has at most a quadratic number of *independent* challenges in d. This reduces to a merely *linear* number of CRPs which the adversary must read out in our attack. Finally, we remark that scaling up the size of the PUF also quickly reaches its limits under practical aspects: The token considered by Pappu et al. [21,20] has an area of 1 cm $\times$ 1 cm. In order to slow down the quadratic attack merely by a factor of 10, a token of area 10 cm $\times$ 10 cm would have to be used. Such a token is too large to even fit onto a smart card.

Overall, this leads to the conclusion that optical PUFs like the ones discussed in [20,21,16] cannot be used safely with the Protocols 1 and 2 in the face of our attack. Due to their low-degree polynomial CRP complexity, and due to practical size constraints, simple scaling of the PUFs constitutes no efficient countermeasure. This distinguishes the optical approach from the electrical case of the last section. This observation has a particular relevance, since Brzuska et al. had explicitly suggested optical PUFs for the implementation of their protocols (see Section 8 of [2]).

[2] Please note in this context that the claim of [2] that the number of CRPs of an optical PUF is super-polynomial must have been made erroneously or by mistake; our above brief analysis shows that it is at mostly cubic. The low-degree polynomial amount of challenges of the optical PUF is indeed confirmed by the entire literature on the topic, most prominently [21,20,31].

5 Potential Countermeasures

5.1 Additional PUF Transfers and Time Constraints?

Can we bind the time in which the malicious player has got access to the PUF in order to prevent our attack? The current Protocols 1 and 2 obviously are unsuited to this end; but could there be modifications of theirs which have this property? A simple approach seems the introduction of one additional PUF transfer from the sender to the receiver in the initialization phase. This assumes that the sender initially holds the PUF, transfers it to the receiver, and measures the time period within which the receiver returns the PUF. The (bounded) period in which the receiver had access to the PUF can then be used to derive a bound on the number of CRPs the receiver might know. This could be used to enforce security against a cheating receiver. Please note that a long, uncontrolled access time for the sender is no problem for the protocol's security, whence it suffices to concentrate on the receiver.

On closer inspection, however, there are significant problems with this approach. In general, each PUF-transfer in a protocol is very costly. One PUF-transfer per protocol seems acceptable, since it is often executed automatically and for free, for example by consumers carrying their bank cards to cash machines. But having two such transfers in one protocol, as suggested above, will most often ruin a protocol's practicality.

A second issue is that binding the adversarial access time *in a tight manner* by two consecutive PUF transfers is very difficult. How long will one physical transfer of the PUF take? 1 day? If the adversary can execute this transfer a few hours faster and can use the gained time for executing measurements on the PUF, our countermeasure fails. The same holds if the adversary carries out the physical transfer himself and can measures the PUF while it is in transit.

In summary, enforcing a tight time bound on the receiver's access time by two PUF transfers or also by other measures will be impossible in almost any applications. The above idea may thus be interesting as a theoretical concept for future PUF-protocol design, but cannot be considered a generally efficient and practically relevant countermeasure.

5.2 Interactive Hashing

Let us now discuss a second and more effective countermeasure: The employment of interactive hashing (IH) as a substep in OT protocols. As we will show, protocols based on IH can achieve better security properties than Protocol 1. The idea of using IH in the context of PUFs has been first been suggested by Rührmair in 2010; his OT-protocol was the first published PUF-based two-player protocol [23]. The following approach is a simplified version of his original scheme. We also give (for the first time) a security analysis of the protocol. Via the general reduction of BC to OT presented in Protocol 2, our construction for OT can also be used to implement PUF-based BC.

5.2.1 Interactive Hashing as a Security Primitive

Interactive hashing (IH) is a two-player security primitive suggested by [18,17]. It has been deployed as a protocol tool in various contexts, including zero-knowledge proofs, bit commitment and oblivious transfer (see references in [17]. The following easily accessible and application-independent definition of IH has been given in [4]; for more a formal treatment see [27].

Definition 4 (Interactive Hashing (IH) [4])**.** *Interactive Hashing is a cryptographic primitive between two players, the sender and the receiver. It takes as input a string $c \in \{0,1\}^t$ from the sender, and produces as output two t-bit strings, one of which is c and the other $c' \neq c$. The output strings are available to both the sender and the receiver, and satisfy the following properties:*

1. *The receiver cannot tell which of the two output strings was the original input. Let the two output strings be c_0, c_1, labeled according to lexicographic order. Then if both strings were a priori equally likely to have been the sender's input c, then they are a posteriori equally likely as well.*
2. *When both participants are honest, the input is equally likely to be paired with any of the other strings. Let c be the sender's input and let c' be the second output of interactive hashing. Then provided that both participants follow the protocol, c' will be uniformly distributed among all $2^t - 1$ strings different from c.*
3. *The sender cannot force both outputs to have a rare property. Let $\mathcal{G}$ be a subset of $\{0,1\}^t$ representing the sender's "good set". Let G be the cardinality of $\mathcal{G}$ and let $T = 2^t$. Then if G/T is small, the probability that a dishonest sender will succeed in having both outputs c_0, c_1 be in $\mathcal{G}$ is comparably "small".*

One standard method to implement IH is by virtue of a classical technique by Naor et al. [17]. To achieve a self-contained treatment, we describe this technique in a variant introduced by Crepeau et al. [4] below. In the protocol below, let c be a t-bit string that is the input to sender in the interactive hashing. All operations take place in the binary field $\mathcal{F}_2$.

Protocol 5: INTERACTIVE HASHING [4]

1. The receiver chooses a $(t-1) \times t$ matrix $\mathbf{Q}$ uniformly at random among all binary matrices of rank $t-1$. Let q_i be the i-th query, consisting of the i-th row of $\mathbf{Q}$.
2. For $1 \leq i \leq t-1$ do:
   (a) The receiver sends query q_i to the sender.
   (b) The sender responds with $v_i = q_i \cdot c$.
   (c) Given $\mathbf{Q}$ and $v \in \{0,1\}^{t-1}$ (the vector of the sender's responses), both parties compute the two values of $c \in \{0,1\}^t$ consistent with the linear system $\mathbf{Q} \cdot c = v$. These solutions are labeled c_0, c_1 according to lexicographic order.

The following theorem, which is taken from [4,27], tells us about the security of the above scheme. It relates to the security definition 4.

Theorem 6 (Security of Protocol 5). *Protocol 5 satisfies all three information theoretic security properties of Definition 4. Specifically, for Property 3 of Definition 4, it ensures that a dishonest sender can succeed in causing both outputs to be in the "good set" $\mathcal{G}$ with probability at most $15.6805 \cdot G/T$, where $G = |\mathcal{G}|$ and $T = 2^t$.*

5.2.2 Oblivious Transfer

We are now presenting a PUF-based oblivious transfer protocol that uses IH as a substep. It bears some similarities with an earlier protocol of Rührmair [23] in the sense that it also uses interactive hashing, but is slightly simpler.

Protocol 7: PUF-BASED 1-OUT-OF-2 OBLIVIOUS TRANSFER WITH INTERACTIVE HASHING

1. The sender's input are two strings $s_0, s_1 \in \{0,1\}^\lambda$ and the receiver's input is a bit $b \in \{0,1\}$.
2. The receiver chooses a challenge $c \in \{0,1\}^\lambda$ uniformly at random. He applies c to the PUF, which responds r. He transfers the PUF to the sender.
3. The sender and receiver execute an IH protocol, where the receiver has input c. Both get outputs c_0, c_1. Let i be the value where $c_i = c$.
4. The receiver sends $b' := b \oplus i$ to the sender.
5. The sender applies the challenges c_0 and c_1 to the PUF. Denote the corresponding responses as r_0 and r_1.
6. The sender sends $S_0 := s_0 \oplus r_{b'}$ and $S_1 := s_1 \oplus r_{1-b'}$ to receiver.
7. The receiver recovers the string s_b that depends on his choice bit b as $S_b \oplus r = s_b \oplus r_{b \oplus b'} \oplus r = s_b \oplus r_i \oplus r = s_b$.

5.2.3 Security and Practicality Analysis

We start by a security analysis of Protocol 7 in the so-called "stand alone, good PUF model", which was introduced by van Dijk and Rührmair in [6]. In this communication model, the following two assumptions are made: (i) the PUF-protocol is executed only once, and the adversary or malicious players have no access to the PUF anymore after the end of the protocol; (ii) the two players do not manipulate the used PUFs on a hardware level. We stress that whenever these two features cannot be guaranteed in practical applications, a number of unexpected attacks apply, which spoil the security of the respective protocols. Even certain impossibility results can be shown under these circumstances; see [6] for details.

In the following analysis in the stand alone, good PUF model, we assume that the adversary has the following capabilities:

1. He knows a certain number of CRPs of the PUF, and has possibly used them to build an (incomplete) predictive model of the PUF. In order to model this ability, we assume that there is a proper subset $S \subsetneq C$ of the set of all challenges C such that the adversary knows the correct responses to the challenges in S with probability one. The cardinality of S depends on the previous access times of the adversary to the PUF and the number of CRPs he has collected from other sources, for example protocol eavesdropping. It must be estimated by the honest protocol users based on the given application scenario. Usually $|S| \ll |C|$.
2. Furthermore, we assume that the adversary can correctly guess the response to a uniformly and randomly chosen challenge $c \in C \setminus S$ with probability at most ϵ, where the probability is taken over the choice of c and over the adversary's

random coins. Usually ϵ will be significantly smaller than one. To name two examples: In the case of a well-designed electrical PUF with single-bit output, ϵ will be around 0.5; in the case of a well-designed optical PUF [20,21] with multi-bit images as outputs, ϵ can be extremely small, for example smaller than 2^{-100}. Again, the honest protocol users must estimate ϵ based on the circumstances and the employed PUF.

Assuming the above capabilities and using Theorem 6, the probability that the receiver can cheat in Protocol 7 is bounded above by

$$15.6805 \cdot |S|/|C| + \epsilon,$$

a term that will usually be significantly smaller than one.

Under the presumption that this cheating probability of the receiver is indeed smaller than one, the security of Protocol 7 can be further amplified by using a well-known result by Damgard, Kilian and Savail (see Lemma 3 of [5]):

Theorem 8 (OT-Amplification [5]). *Let (p, q)-WOT be a 1-2-OT protocol where the sender with probability p learns the choice bit c and the receiver with probability q learns the other bit b_{1-c}. Assume that $p + q < 1$. Then the probabilities p and q can be reduced by running k (p, q)-WOT-protocols to obtain a $(1 - (1 - p)^k, q^k)$-WOT protocol.*

In the case of our OT-Protocol 7 it holds that $p = 0$, whence the technique of Damgard et al. leads to an efficient security amplification, and to a $(0, q^k)$-WOT protocol. The PUF does not need to be transferred k times, but one PUF-transfer suffices. We remark that the probability amplification according to Theorem 8 is not possible with Protocol 1 after our quadratic attack, since the attack leads to a cheating probability of one for the receiver, i.e., $p + q \geq 1$ in the language of Theorem 8.

Let us quantatively illustrate the security gain of Protocol 7 over Protocol 1 via a simplified back-of-the-envelope calculation: We argued earlier that via our quadratic attack, a malicious receiver who has read out $2 \cdot 2^{18}$ CRPs from an optical PUF can cheat with probability 1 (= with certainty) in Protocol 1. Let us compare this to the case that an optical PUF is used in the IH-based Protocol 7. Let us assume that the adversary has collected the same number of CRPs ($= 2 \cdot 2^{18}$ CRPs) as in the quadratic attack, and that the (multi-bit) response of the optical PUF on the remaining CRPs is still hard to preduct, i.e., it cannot be predicted better than with probability $\epsilon \leq 2^{-100}$. Then by Theorem 6 and by our above analysis, the adversary's chances to break Protocol 7 are merely around $15.6805 \cdot 2^{19} \cdot 2^{-35} + 2^{-100} \approx 0.00024$. This probability can then be exponentially reduced further via Theorem 8.

On the downside, however, the IH-Protocol 5 has a round complexity that is linear (i.e., equal to $\lambda - 1$) in the security parameter λ. This is relatively significant for the optical PUF (where $\lambda = 35$) and electrical PUFs with medium bitlengths (where $\lambda = 64$). One possible way to get around this problem is to use the constant round interactive hashing scheme by Ding et al. [7]. However, this scheme has slightly worse security guarantees than the IH scheme of the last sections. Future work will analyze the exact security loss under the use of the IH scheme of Ding. A first analysis to this end can be found in van Dijk and Rührmair [6].

To summarize the discussion in this section, interactive hashing can restore the security of PUF-based OT protocols even for small sized PUFs with 64-bit challenge lengths and for optical PUFs in the stand alone, good PUF model. Via the general reduction of BC to OT given in Protocol 2, this result can be used to securely implement PUF-based BC in this model, too. However, the use of IH leads to an increased number of communication rounds that is about equal to the (binary) challenge length of the PUF, i.e., around 64 rounds for the integrated PUFs with 64 bit challenges, and around 35 rounds for optical PUFs of size 1 cm^2 [21]. It must be decided on the basis of the concrete application scenario whether such a number of rounds is acceptable.

6 Summary and Conclusions

We revisited PUF-based OT- and BC-protocols, including the recent schemes of Rührmair from Trust 2010 [23] and Brzuska et al. from Crypto 2011 [1,2]. We placed special emphasis on the security which these protocols achieve in practice, in particular when they are used in connection with widespread optical and 64-bit electrical PUF-implementations. Our analysis revealed several interesting facts.

First of all, we described a simple and efficient method by which the OT- and BC-protocol of Brzuska et al. can be attacked with probability one in practice if electrical PUFs with 64-bit challenge lengths are used, or whenever optical PUFs are employed. Since much research focuses on 64-bit implementations of electrical PUFs [9,10,15], and since Brzuska et al. had explicitly suggested optical PUFs for the implementation of their protocols (see Section 8 of [2]), the publication and dissemination of our quadratic attack seems important to avoid their use in Protocols 1 and 2. Please note that our attack is independent of the cryptographic hardness of the PUF, and is merely based on its challenge size.

Secondly, we discussed an alternative class of protocols for oblivious transfer that are based on interactive hashing techniques. They are inspired by the earlier OT-protocol of Rührmair [23]. We argued that these protocols lead to better security in practice. They can be used safely with 64-bit electrical PUFs. When used with optical PUFs, they lead to better security than the protocols of Brzuska et al., but the security margins are tighter than in the 64-bit case. In both cases, a well-known result by Damgard, Kilian and Savail [5] can be used in order to reduce the cheating probabilities exponentially.

Our discussion shows once more that PUFs are quite special cryptographic and security tools. Due to their finite nature, asymptotic constants that might usually be hidden in $O(\cdot)$- and $\Theta(\cdot)$-notations become relevant in practice and should be discussed explicitly. Furthermore, their specific nature often allows new and unexpected forms of attacks. One of the aims of our work is to bridge the gap between PUFs in theory and applications; reconciling these two fields seems a necessary prerequisite for a healthy long-term development of the field. We hope that the general methods and the approach of this paper can contribute to this goal.

Recommendations for Protocols Use and Future Work. Let us conclude the paper with a condensed recommendation for the practical implementation of PUF-based OT and BC protocols, and by a discussion of future work. Firstly, it is clear from our results that the protocol of Brzuska et al. cannot be used safely with optical PUFs a la Pappu (i.e., with non-integrated optical PUFs that have only a small or medium sized challenge set), or with electrical PUFs with challenge lengths around 64 bits.

Secondly, we showed that Protocols based on interactive hashing (IH) can achieve better security. These protocols can be employed safely with optical PUFs and with electrical PUFs of challenge length 64. Furthermore, Damgard et al.'s [5] amplification technique can be applied in order to bring the cheating probabilities arbitrarily close to zero. Nevertheless, we would like to stress once more to practical PUF-users that this analysis only applies if the protocols are employed in the stand alone, good PUF model (see Section 5.2.3 and [6]). As soon as the features of this model cannot be enforced in a given application (for example by certifying a PUF, or by erasing PUF responses at the protocol end [6]), certain new attacks apply, which spoil both the security of IH-based protocols and of the protocols of Brzuska et al. These attacks are not the topic of this publication, but have been described in all detail in [6].

If a PUF has challenge length of 128 bits or more, it seems at first sight that the protocols of Brzuska et al. could be used safely *in the stand alone, good PUF model, too,* but we stress that this recommendation is yet to be confirmed by full formal analysis. One issue is that the PUF security feature required by the protocols of Brzuska et al. is (in a nutshell) that the adversary must be unable to select two PUF-challenges with a given distance d such that he knows the two corresponding responses. This security property of a PUF is new in the literature and should yet be further investigated in future work before final recommendations are being made. In particular, it does not seem simple or straightforward to judge in practice whether a given PUF fulfills this property.

A second topic for future research is how the round complexity of the IH-based protocols can be reduced. Some steps to this end have been made by van Dijk and Rührmair in [6], where the constant-round interactive hashing scheme of Ding et al. [7] is applied to obtain contant-round PUF-based OT and BC protocols.

Acknowledgements. The authors would like to thank Stefan Wolf and Jürg Wullschleger for enjoyable discussions, and Stefan Wolf for suggesting the example in Section 3, page 256 to us. Part of this work was conducted within the physical cryptography project at the TU München.

References

1. Brzuska, C., Fischlin, M., Schröder, H., Katzenbeisser, S.: Physically Uncloneable Functions in the Universal Composition Framework. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 51–70. Springer, Heidelberg (2011)
2. Brzuska, C., Fischlin, M., Schröder, H., Katzenbeisser, S.: Physical Unclonable Functions in the Universal Composition Framework. Full version of the paper. Available from Cryptology ePrint Archive (2011) (downloaded on February 28, 2012)
3. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: FOCS 2001, pp. 136–145 (2001); Full and updated version available from Cryptology ePrint Archive
4. Crépeau, C., Kilian, J., Savvides, G.: Interactive Hashing: An Information Theoretic Tool (Invited Talk). In: Safavi-Naini, R. (ed.) ICITS 2008. LNCS, vol. 5155, pp. 14–28. Springer, Heidelberg (2008)

5. Damgård, I., Kilian, J., Salvail, L.: On the (Im)possibility of Basing Oblivious Transfer and Bit Commitment on Weakened Security Assumptions. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 56–73. Springer, Heidelberg (1999)
6. van Dijk, M., Rührmair, U.: Physical Unclonable Functions in Cryptographic Protocols: Security Proofs and Impossibility Results. Cryptology ePrint Archive, Report 228/2012 (2012)
7. Ding, Y.Z., Harnik, D., Rosen, A., Shaltiel, R.: Constant-round oblivious transfer in the bounded storage model. Journal of Cryptology 20(2), 165–202 (2007)
8. Gassend, B.: Physical Random Functions. MSc Thesis. MIT (2003)
9. Gassend, B., Clarke, D.E., van Dijk, M., Devadas, S.: Silicon physical random functions. In: ACM Conference on Computer and Communications Security 2002, pp. 148–160 (2002)
10. Gassend, B., Lim, D., Clarke, D., van Dijk, M., Devadas, S.: Identification and authentication of integrated circuits. Concurrency and Computation: Practice & Experience 16(11), 1077–1098 (2004)
11. Guajardo, J., Kumar, S.S., Schrijen, G.-J., Tuyls, P.: FPGA Intrinsic PUFs and Their Use for IP Protection. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 63–80. Springer, Heidelberg (2007)
12. Impagliazzo, R., Rudich, S.: Limits on the Provable Consequences of One-Way Permutations. In: STOC 1989, pp. 44–61 (1989)
13. Kilian, J.: Founding cryptography on oblivious transfer. In: STOC (1988)
14. Kumar, S.S., Guajardo, J., Maes, R., Schrijen, G.J., Tuyls, P.: The Butterfly PUF: Protecting IP on every FPGA. In: HOST 2008, pp. 67–70 (2008)
15. Lee, J.-W., Lim, D., Gassend, B., Suh, G.E., van Dijk, M., Devadas, S.: A technique to build a secret key in integrated circuits with identification and authentication applications. In: Proceedings of the IEEE VLSI Circuits Symposium (June 2004)
16. Maes, R., Verbauwhede, I.: Physically Unclonable Functions: a Study on the State of the Art and Future Research Directions. In: Naccache, D., Sadeghi, A.-R. (eds.) Towards Hardware-Intrinsic Security, sec. 1. Springer (2010)
17. Naor, M., Ostrovsky, R., Venkatesan, R., Yung, M.: Perfect zero- knowledge arguments for NP using any one-way permutation. Journal of Cryptology (1998); Preliminary version In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 196–214. Springer, Heidelberg (1993)
18. Ostrovsky, R., Venkatesan, R., Yung, M.: Fair games against an all-powerful adversary. In: AMS DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pp. 155–169 (1993); Preliminary version in SEQUENCES 1991
19. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Lightweight Secure PUFs. In: IC-CAD 2008, pp. 607–673 (2008)
20. Pappu, R.: Physical One-Way Functions. PhD Thesis, Massachusetts Institute of Technology (2001)
21. Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical One-Way Functions. Science 297, 2026–2030 (2002)
22. Rivest, R.: Illegitimi non carborundum. Invited keynote talk, CRYPTO 2011 (2011)
23. Rührmair, U.: Oblivious Transfer Based on Physical Unclonable Functions. In: Acquisti, A., Smith, S.W., Sadeghi, A.-R. (eds.) TRUST 2010. LNCS, vol. 6101, pp. 430–440. Springer, Heidelberg (2010)
24. Rührmair, U., Busch, H., Katzenbeisser, S.: Strong PUFs: Models, Constructions and Security Proofs. In: Sadeghi, A.-R., Tuyls, P. (eds.) Towards Hardware Intrinsic Security: Foundation and Practice. Springer (2010)
25. Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., Schmidhuber, J.: Modeling Attacks on Physical Unclonable Functions. In: ACM Conference on Computer and Communications Security (2010)

26. Rührmair, U., Sölter, J., Sehnke, F.: On the Foundations of Physical Unclonable Functions. Cryptology e-Print Archive (June 2009)
27. Savvides, G.: Interactive Hashing and reductions between Oblivious Transfer variants. PhD thesis, McGill University, Montreal (2007)
28. Suh, G.E., Devadas, S.: Physical Unclonable Functions for Device Authentication and Secret Key Generation. In: DAC 2007, pp. 9–14 (2007)
29. Tuyls, P., Schrijen, G.-J., Škorić, B., van Geloven, J., Verhaegh, N., Wolters, R.: Read-Proof Hardware from Protective Coatings. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 369–383. Springer, Heidelberg (2006)
30. Tuyls, P., Škorić, B.: Strong Authentication with Physical Unclonable Functions. In: Petkovic, M., Jonker, W. (eds.) Security, Privacy and Trust in Modern Data Management. Springer (2007)
31. Tuyls, P., Škorić, B., Stallinga, S., Akkermans, A.H.M., Ophey, W.: Information-Theoretic Security Analysis of Physical Uncloneable Functions. In: Patrick, A.S., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 141–155. Springer, Heidelberg (2005)

Chapter 5

An Attack on PUF-based Session Key Exchange and a Hardware-Based Countermeasure: Erasable PUFs

One of the earliest cryptographic protocols for Strong PUFs that went beyond simple identification is a combined authentication and session key exchange method by Tuyls and Skoric [86] from 2007. It is designed for a PUF that is contained on a bank card and used in several automated teller machines (ATMs) on the one side, and the bank headquarters or bank server on the other side. In practical applications, is natural to assume that this protocol is executed multiple times in various ATMs. This implies that adversaries may gain access to the PUF several times and between the execution of different protocol sessions. Since a Strong PUF has a publicly readable, unprotected challenge-response-interface (see Chapter 2), this allows adversaries to read out PUF-CRPs of their choice on multiple occasions.

We show in this chapter that the protocol is not secure in such natural settings, and that already twofold access allows adversaries to derive earlier session keys. Subsequently, we turn to potential countermeasures. How could the security problems arising from multiple adversarial access be overcome? We argue briefly that such countermeasures must likely be found on the hardware side (i.e., on the side of the PUF) and not on the protocol side. Next, we suggest a new PUF-primitive that possesses the necessary features to thwart our attack, namely so-called *"Erasable PUFs"*. These are Strong PUFs with the additional feature that arbitrary single responses can be erased without affecting any other responses.

Besides identifying and defining this new primitive, we also make first steps towards its implementation: We present real measurement data from random silicon diodes, so-called “ALILE diodes”, which possess a particularly large variation in the I(V)-curves compared to other Strong PUFs. The measurement data shows that the random information in the diodes can be changed or “erased” by applying a current above a certain threshold, while not completely burning and destroying the diode, at least rudimentarily maintaining its functionality.

The paper used in this chapter is

- U. Rührmair, C. Jaeger, M. Algasinger: *An Attack on PUF-Based Session Key Exchange and a Hardware-Based Countermeasure: Erasable PUFs*. Financial Cryptography and Data Securiy (FC 2011), Lecture Notes in Computer Science, Vol. 7035, pp. 190-204, Springer, 2012.

As an outlook, we remark that the observation that Strong PUF protocols may be vulnerable under multiple PUF access later will lead to the formal introduction of the so-called *“PUF re-use model”* in Chapter 6; the current Chapter 5 prepares the stage for this follow-up work.

According to Google scholar, the work underlying this chapter has been cited 29 times to this date [26].

An Attack on PUF-based Session Key Exchange and a Hardware-based Countermeasure: Erasable PUFs

Ulrich Rührmair [†] [*], Christian Jaeger [◇], and Michael Algasinger [◇]

[†]Computer Science Department
[◇]Walter Schottky Institut
Technische Universität München
85748 Garching
ruehrmair@in.tum.de,
{christian.jaeger,michael.algasinger}@wsi.tum.de
http://www.pcp.in.tum.de

Abstract. We observe a security issue in protocols for session key exchange that are based on Strong Physical Unclonable Functions (PUFs). The problem is illustrated by cryptanalyzing a recent scheme of Tuyls and Skoric [1], which has been proposed for use in a bank card scenario. Under realistic assumptions, for example that the adversary Eve can eavesdrop the communication between the players and gains physical access to the PUF twice, she can derive previous session keys in this scheme. The observed problem seems to require the introduction of a new PUF variant, so-called *"Erasable PUFs"*. Having defined this new primitive, we execute some first steps towards its practical implementation, and argue that Erasable PUFs could be implemented securely via ALILE-based crossbar structures.

1 Introduction

Motivation and Background. Electronic devices have pervaded our everyday life, making them a well-accessible target for adversaries. Classical cryptography offers several measures against the resulting security and privacy problems, but they all rest on the concept of a secret binary key: They presuppose that the electronic devices can contain a piece of information that is, and remains, unknown to an adversary. However, this requirement can be difficult to uphold in practice: Physical attacks such as invasive, semi-invasive, or side-channel attacks, as well as software attacks like API-attacks and viruses, can lead to key exposure and security breaks.

The described situation was one motivation that led to the development of *Physical Unclonable Functions (PUFs)*. 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 R_i. Contrary to standard digital systems, a PUF's responses shall depend on the nanoscale structural disorder present in it. 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. This means that any PUF S implements

[*] Corresponding author.

an individual function F_S mapping challenges C_i to responses R_i. The tuples (C_i, R_i) are thereby often called the *challenge-response pairs (CRPs)* of the PUF.

Due to its complex internal structure, a PUF can avoid some of the shortcomings associated with digital keys. It is usually harder to read out, predict, or derive its responses than to obtain the values of digital keys stored in non-volatile memory. This fact has been exploited for various PUF-based security protocols. Prominent examples include schemes for identification [2], key exchange [1], or digital rights management purposes [3] [5]. Another advantage of (Strong) PUFs is that they can lead to protocols whose security does not depend on the usual, unproven number theoretic assumptions (such as the factoring or discrete logarithm problem), but rests on independent hypotheses.

Strong PUFs and Weak PUFs. Two important subtypes of PUFs, which must explicitly be distinguished in this paper, are *Strong PUFs* [1] and *Weak PUFs* [2]. This distinction has been made first in [4] [3], and has been elaborated on further in [6] [7] [8].

Strong PUFs are PUFs with a very large number of possible challenges. The adversarial ability to apply challenges to them and to read out their responses from them is usually not restricted. Their central security features are: (i) It must be impossible to physically clone a Strong PUF, i.e. to fabricate a second system which has the same challenge-response-behavior as the original PUF. This restriction must hold even for the original manufacturer of the PUF. (ii) Due to the very large number of possible challenges and the PUF's finite read-out rate, a complete measurement of all challenge-response pairs (CRPs) within a limited time frame (such as several days or even weeks) must be impossible. (iii) It must be difficult to numerically predict the response R_i of a Strong PUF to a randomly selected challenge C_i, even if many other challenge-response pairs are known.

A complete formal specification of Strong PUFs is laborious and besides the scope of this paper, but can be found in [7]. Examples of candidates for Strong PUFs are complex optical scatterers [2] or special, delay-based integrated circuits [9] [10] [11] (albeit several of the latter have been broken up to a certain size in recent machine learning attacks [6]). Also analog circuits have been proposed recently [12].

Weak PUFs may have very few challenges — in the extreme case just one, fixed challenge. Their response(s) R_i are used to derive a standard secret key, which is subsequently processed by the embedding system in the usual fashion, e.g. as a secret input for some cryptoscheme. Contrary to Strong PUFs, the responses of a Weak PUF are never meant to be given directly to the outside world.

Weak PUFs essentially are a special form of non-volatile key storage. Their advantage is that they may be harder to read out invasively than non-volatile memory like

[1] Strong PUFs have also been referred to as Physical Random Functions [9] [10], or (almost equivalently) as Physical One-Way Functions [2] in the literature.

[2] Weak PUFs have also been referred to as Physically Obfuscated Keys (POKs) [4]. Note that the predicate "Weak" is not meant to state that these PUFs are "bad" in any sense, we just follow the terminology introduced in [3].

EEPROM. Typical examples of Weak PUFs are the SRAM PUF [3], Butterfly PUF [5] and Coating PUF [13].

Applications of Strong PUFs. We are mostly concerned with Strong PUFs and variants thereof in this paper, whence we focus on them from now on. The archetypical application of Strong PUFs is the identification of entities over insecure networks. It has already been suggested in the first PUF publication [2] by the example of a bank card scenario, and works along the following lines. Each customer's bank card contains an individual Strong PUF. Before issuing the card, the bank measures several of the PUF's CRPs, and stores them secretly on its server. When the customer inserts his card into a terminal, the terminal contacts the bank. The bank chooses at random several challenges C_i from its secret CRP list, and sends them to the terminal. The terminal obtains the corresponding responses R_i from the PUF, and returns them to the bank. If they match the values in the CRP list, the bank considers the card as genuine. The scheme has the upsides of circumventing the need for secret keys or secret information on the vulnerable bank cards, and of avoiding the usual, unproven complexity theoretic assumptions of classical identification protocols.

A second, central application of Strong PUFs, which also has already been suggested in [2] (page 2029), and which has been worked out in greater detail in [1], is the distribution of a secret key between different parties, for example the terminal and the bank. We are mainly concerned with this second application in this paper.

Our Contributions. Our first contribution is to observe a problem in the repeated use of PUF-based session key exchange protocols. We illustrate this problem by the example of a recent protocol by Tuyls and Skoric [1], which has originally been suggested for use in a bank card scenario. We show how to cryptanalyze this protocol under the presumptions that an adversary can eavesdrop the communication between the terminal and the bank, that he has got access to the PUF more than once, and that no secret digital information can be stored on the card. These presumptions seem very natural, even more so in the original application scenario of bank cards or credit cards (see section 2). The problem which our attack exploits is that the CRP-information used to derive a key remains present in the PUF after the completion of the key exchange protocol.

Second, we reason that the described problem cannot be solved via protocol or software measures, and also not on the basis of current PUF architectures. Resolution seems to require the introduction of a new PUF variant, so-called Erasable PUFs. They are a special type of Strong PUF, with the additional feature that the value of single responses can be erased or altered without affecting the value of all other responses. We specify this new primitive, and show how it can be used to fix the above security issues.

Third, we suggest one possible implementation strategy for Erasable PUFs: Large, monolithic crossbar arrays of diodes with random current-voltage characteristics. It has already been demonstrated in earlier work that such crossbar arrays can act as secure Strong PUFs [14] [15] [16]. We now show that the information stored in the diodes of the crossbar can be erased individually: By applying dedicated voltage pulses to selected crossbar wires, the current-voltage curve of any single diode can be altered individually, and without affecting the other diodes in the array. We present measurement data from single ALILE-diodes fabricated in our group that supports the feasibility of the described approach.

Related Work. There is no related work concerning the cryptanalysis of the Strong PUF-based session key exchange protocol by Tuyls and Skoric. In general, the cryptanalysis of PUF-based protocols appears to be a relatively recent field. Previous PUF attacks mainly focused on breaking the security properties of PUFs themselves (for example by modeling Strong PUFs via machine learning techniques [6]), but not on analyzing PUF protocols.

With respect to Erasable PUFs, there is obviously a large body of work on Strong PUFs and Weak PUFs, but none of them explicitly considered the property of erasing individual CRPs without affecting other CRPs. The category of PUFs which comes closest to Erasable PUFs are Reconfigurable PUFs (r-PUFs) [17], but the previously proposed optical, scattering-based implementation of r-PUFs has the property that inevitably *all* CRPs are altered by the reconfiguration operation. No erasure or alteration on a single CRP level is enabled. See also section 4 for a further discussion.

Organization of the Paper. In Section 2, we illustrate a security problem occurring in PUF-based key establishment protocols. Section 4 discusses the implementation of Erasable PUFs via crossbar structures. Section 4 describes a few obstacles in the practical realization of Erasable PUFs. Section 5 gives some background on the recent concept of a Crossbar PUF. Section 6 describes how information can be erased from Crossbar PUFs, implementing Erasable PUFs. We conclude the paper in Section 7.

2 A Problem with PUF-based Session Key Establishment

2.1 The Protocol of Tuyls and Skoric

A specific Strong PUF-based protocol for combined identification and session key establishment has been suggested recently in [1]. It is illustrated in Fig. 1. The protocol is run between a Bank on the one hand and an Automated Teller Machine (ATM) plus a security token carrying the Strong PUF on the other hand. It presumes that all involved parties have knowledge of a public one-way hash function h, and of a publicly known error correction scheme, which is used to derive secrets S from a given noisy PUF-response R and helper data W.

The protocol presupposes a set-up phase, in which the bank has got direct access to the Strong PUF. The bank first of all establishes a (large) secret list of the form $\{C_i, W_i, S'_i\}$. Thereby the C_i are randomly chosen challenges, W_i denotes helper data that is generated by the bank from the corresponding (noisy) responses R_i of the PUF, and S'_i refers to secret information that is derived from the noisy response by use of the helper data. Furthermore, the bank chooses a secret numerical value x at random, and writes $h(x)$ onto the card. After that, the card is released to the field.

Each subsequent execution of the protocol is run between the bank and the ATM/PUF. At the beginning of the protocol, the token stores the number n of previous protocol executions, the value $m = h^n(x)$, and an identification number of the Strong PUF, denoted as ID_{PUF}.

The Bank initially holds a list of the form $\{C_i, W_i, S_i'\}$ that is stored together with ID_{PUF} in the Bank's database. The value n' says how often the Bank has been engaged in a session key exchange protocol with the PUF, and $m' = h^{n'}(x)$. The rest of the protocol is described in Fig. 1, which is essentially taken from [1]. At the end of the protocol, the Bank and the ATM/PUF have established a joint session key K.

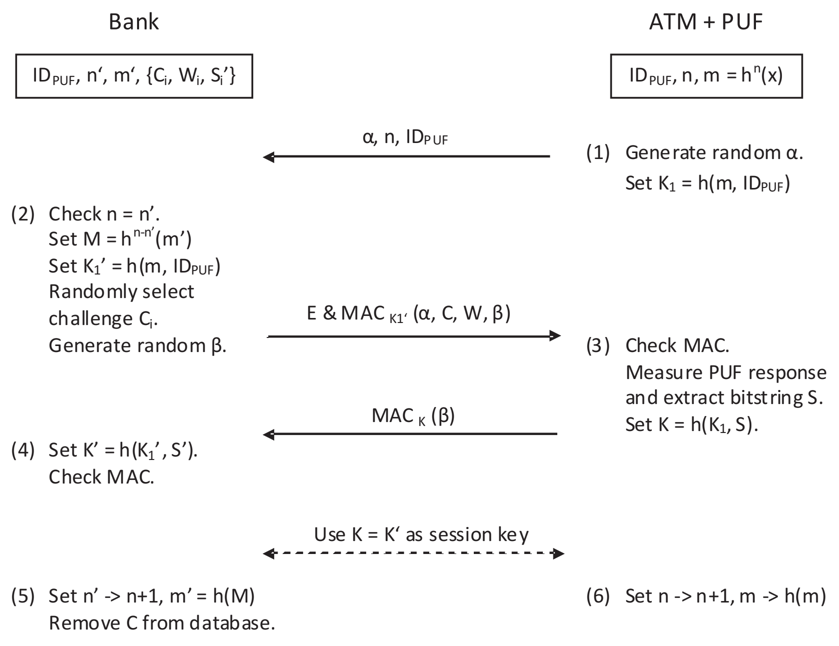

Bank
ATM + PUF
ID$_{PUF}$, n', m', {C$_i$, W$_i$, S$_i$'}
ID$_{PUF}$, n, m = h^n(x)
α, n, ID$_{PUF}$
(1) Generate random α. Set K$_1$ = h(m, ID$_{PUF}$)
(2) Check n = n'. Set M = h$^{n-n'}$(m') Set K$_1$' = h(m, ID$_{PUF}$) Randomly select challenge C$_i$. Generate random β.
E & MAC$_{K1'}$ (α, C, W, β)
(3) Check MAC. Measure PUF response and extract bitstring S. Set K = h(K$_1$, S).
MAC$_K$ (β)
(4) Set K' = h(K$_1$', S'). Check MAC.
Use K = K' as session key
(5) Set n' -> n+1, m' = h(M) Remove C from database.
(6) Set n -> n+1, m -> h(m)

Fig. 1. A protocol for combined identification and session key exchange based on Strong PUFs, which has been suggested by Tuyls and Skoric in [1].

2.2 Problems Arising from Repeated Access to the PUF

We will now present an attack on the repeated use of the above protocol, which allows Eve to derive previous session keys.

The attack makes the following assumptions: (A) Eve can eavesdrop the communication between the bank and the ATM/PUF. (B) No secret digital numbers (e.g., hash values, secret keys) can be stored safely in a non-volatile fashion on the security token. (C) Eve gains access to the security token at least twice, and can measure selected CRPs from the Strong PUF on the token. All of these assumptions are relatively well motivated: If a secure channel would be at hand, which cannot be eavesdropped by Eve, then no complicated session key exchange protocol is necessary. The secret keys could simply be exchanged by sending them over this channel. Likewise, if we were to assume that secret digital keys (or other secret digital numbers) could be stored safely on the token, then the use of PUFs is unnecessary: The token could execute all necessary communication securely via classical, secret key based cryptography. Finally, assumption (C) is straightforward: For example in a bank card scenario, where an adversary might operate with faked terminals/readers that are under his control, and where the card is inserted multiple times into these terminals/readers. Again, if we do not allow an adversary to obtain physical access to the card, then the use of PUFs is unnecessary in the first place.

Eve's attack works in three successive phases executed at times T_1, T_2 and T_3.[3] In the first phase at time T_1, we presume that Eve has got access to the token according to assumption (C). By assumption (B), she can read the current values of n and m at time T_1 from the token, denoted by $n(T_1)$ and $m(T_1)$.

In the second attack phase at time T_2, we assume that Eve eavesdrops a session key establishment protocol between the bank and the ATM/PUF. This is possible according to assumption (A). From the first message sent in the protocol, which we denote by $\alpha(T_2), n(T_2), ID_{PUF}$, Eve learns the current counter value $n(T_2)$. Since Eve already knows $n(T_1)$ and $m(T_1)$ from phase 1, she can deduce the current state $m(T_2) = h^{n(T_2)}(x) = h^{n(T_2)-n(T_1)}(m(T_1))$. This allows her to derive the value of the preliminary key K_1 at time T_2 by setting $K_1(T_2) = h(m(T_2), ID_{PUF})$. Now, when the bank sends the protocol message $E\&MAC_{K_1'(T_2)}(\alpha(T_2), C(T_2), W(T_2), \beta(T_2))$, Eve can remove the encryption, because she knows $K_1(T_2) = K_1'(T_2)$. She learns $C(T_2)$ and the helper data $W(T_2)$. This closes Eve's contribution in the second attack phase. In the further course of the protocol (and without Eve's involvement), the ATM/PUF measures the PUF and extracts a secret bitstring $S(T_2)$ from its responses. Finally, the ATM/PUF sets the session key to be $K(T_2) = h(K_1(T_2), S(T_2))$.

In the third attack phase at time T_3, we assume that Eve has got access to the security token and the Strong PUF, and that she can measure CRPs of the Strong PUF. This is in accordance with assumption (C). Eve uses this ability to measure the PUF's responses $R(T_2)$ that correspond to the challenge(s) $C(T_2)$. Note that the Strong PUF's responses are time invariant and are not actively altered by any protocol participant. Hence Eve can determine $R(T_2)$, even though the time has progressed to T_3 at this point. Eve also knows $W(T_2)$, whence she can derive $S(T_2)$ from the responses $R(T_2)$. This enables her to compute $K(T_2) = h(K_1(T_2), S(T_2))$, since she knows $K_1(T_2)$ already. In other words, Eve obtains the session key $K(T_2)$ that was derived and used at time T_2, breaking the protocol's security.

2.3 Consequences for CRP Refreshment and Identification

It has been suggested in [1] that a session key K established via the protocol of Fig. 1 could be used to achieve CRP refreshment between the ATM and the Bank. To that

[3] In the description of our attack, we will need to consider the value of various protocol parameters, such as n, m, or K_1, at different points in time. To avoid confusion, we use the notation $n(T)$, $m(T)$, $K_1(T)$ (or similar expressions) to denote the values of n, m or K_1 at time T.

end, the ATM would, in regular intervals, execute the following steps: (i) Measure new data of the form $\{C_i(T_j), W_i(T_j), S'_i(T_j)\}$ (where T_j can be an arbitrary point in time). (ii) Exchange a session key $K(T_j)$ via the protocol of Fig. 1. (iii) Send the encrypted message $E\&MAC_{K(T_j)}\{C_i(T_j), W_i(T_j), S'_i(T_j)\}$ to the Bank. (iv) The Bank decrypts this message, and adds $\{C_i(T_j), W_i(T_j), S'_i(T_j)\}$ to its CRP list. This process is termed CRP refreshment. This method allows shorter CRP lists and saves storage requirements on the bank.

But in the attack scenario described in section 2.2, i.e. under the provisions (A) to (C), Eve can break this scheme. First, she can apply the attack described in section 2.2 to obtain $K(T_j)$. She can then decrypt the message $E\&MAC_{K(T_j)}\{C_i(T_j), W_i(T_j), S'_i(T_j)\}$, and hence learns the values $\{C_i(T_j), W_i(T_j), S'_i(T_j)\}$ that were intended for CRP refreshment. This enables her to impersonate the PUF in subsequent identification protocols that are built on these CRP values. For example, it allows her to build faked bank cards.

2.4 Generality and Difficulty of the Problem

The problem we observed in the previous sections does not only apply to the protocol of Fig. 1. It could be argued that any PUF-based protocol for key establishment between a central authority and decentral principals (terminals, hardware, etc.) involves, explicitly or implicitly, the basic procedure that is shown in Fig. 2.

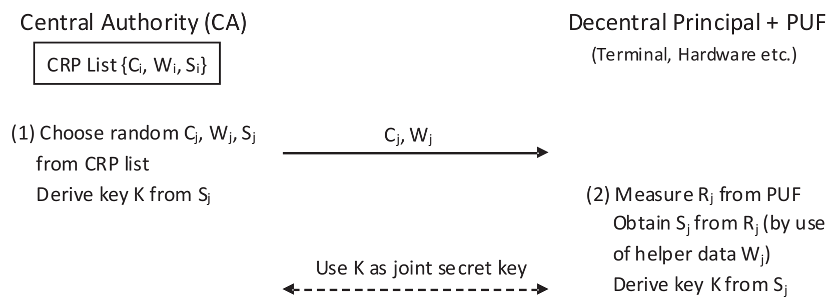

Central Authority (CA)
CRP List {Ci, Wi, Si}
Decentral Principal + PUF
(Terminal, Hardware etc.)
(1) Choose random Cj, Wj, Sj from CRP list Derive key K from Sj
Cj, Wj
(2) Measure Rj from PUF Obtain Sj from Rj (by use of helper data Wj) Derive key K from Sj
Use K as joint secret key

Fig. 2. The "raw", basic building block for PUF-based key exchange. In practice, it can and will usually be accompanied by other measures, such as message authentication or authentication of the physically transferred PUF.

Any protocol of this form is prone to the type of attack described in section 2.2. Considering the protocol of Fig. 2 sheds light on the heart of the problem: Eve can break the protocol by firstly eavesdropping the C_j, W_j. Subsequent one-time access to the PUF allows her to measure the corresponding R_j and to derive the corresponding S_j. This enables her to obtain K. We will not give a full formal proof of this statement, but believe that adapted variants of this simple attack can be mounted on any Strong PUF-based session key exchange. One example for such an adapted attack on a much more complicated protocol was given in Sec. 2.2. The key issue in all cases seems that the response information used for the derivation of K is still extractable from the Strong PUF at later points in time.

It would be hence necessary to "erase" the responses R_j from the Strong PUF after they have been used for key derivation. Note that in such an "erasure" operation, all other responses R_i (with $i \neq j$) must remain unchanged: If they were altered, the list $\{C_i, W_i, S_i\}$ stored at the central authority would no longer be valid. It could neither be used for further key establishment protocols of the above type, nor for the typical PUF-based identification schemes (see Sec. 1).

3 Erasable PUFs

We will now make some first steps work towards a hardware-based solution of the above security problem, introducing a new variant of Strong PUFs: So-called Erasable PUFs. For reasons of clarity and unambiguity, we slightly deviate from the established notation for PUFs in the following specification, and denote the response of a PUF S to a challenge C by R_C^S.

Specification 1 (ERASABLE PUFS)**.** *A physical system S is called an* ERASABLE PUF *if it is a Strong PUF with the following additional properties:*

- *There is a special, physical erasure operation* $\mathbf{ER}(\cdot)$. *It takes as input a challenge C_0 of S. It turns S into a system S' with the following properties:*
  - *S' has got the same set of possible challenges as S.*
  - *For all challenges $C \neq C_0$, it holds that $R_C^{S'} = R_C^S$.*
  - *Given S' and C_0, it is impossible to determine $R_{C_0}^S$ with a probability that is substantially better than random guessing.*

Note that Specification 1 is not meant to be a full-fledged formal definition, but shall mainly express the properties of Erasable PUFs in a compact, semi-formal manner. Its style follows [7].

Given the discussion of the previous sections, it is now relatively straightforward to fix the security issues of the protocols of Fig. 1 and 2.

1. PROTOCOL OF FIG. 1: Use an Erasable PUF in the protocol, and add the erasure operation $\mathbf{ER}(C)$ at the end of step (3).
2. PROTOCOL OF FIG. 2: Use an Erasable PUF in the protocol, and add the erasure operations $\mathbf{ER}(C_j)$ to the end of step (2).

These steps disable the attacks that have been presented in the previous sections: When Eve has got access to the PUF at a later point in time, she can no more determine the PUF responses used for previous key derivation, as the responses have been erased from the system.

4 Obstacles in the Implementation of Erasable PUFs

The implementation of Erasable PUFs on the basis of established PUF architectures turns out to be intricate; we will summarize the occurring difficulties in this section. One reason for the appearing problems is that Erasable PUFs must combine the following properties:

(i) They must be Strong PUFs, i.e. they must have very many possible challenges, and must be able to withstand full read-out for long time periods, i.e. weeks or months.
(ii) They must allow the erasure or alteration of single responses, without affecting other responses.

These properties rule out Weak PUFs and their current implementation candidates [3] [5] [13] from the start, since they simply do not fulfill condition (i) above, i.e. they are no Strong PUFs.

An alternative approach would be to modify Strong PUF architectures in order to obtain Erasable PUFs. The erasure operation could, for example, alter some internal components of a Strong PUF. But unfortunately, all popular candidates for Strong PUFs [2] [9] [10] [11] [12] create their responses in a complex interplay of many or even all internal components. Altering one single component will not only change a single response, but will affect many other responses, too. Their responses cannot be altered individually, i.e. with single CRP granularity.

Another, straightforward idea would be to attach an access control module to a Strong PUF. The module could store a list of "forbidden" challenges and prevent the application of these challenges to the Strong PUF. But this approach is costly in practice: It requires non-volatile memory, which must store potentially large amounts of challenges. Furthermore, it cannot reach ultimate security levels: The control module might be circumvented or cut off by a well-equipped attacker, and the content of the memory (i.e. the forbidden challenges) might be manipulated.

The existing concept that presumably comes closest to Erasable PUF are Reconfigurable PUFs (r-PUFs), which were introduced in [17]. By definition, each r-PUF possesses a reconfiguration operation, in which all CRPs of the r-PUF can be changed. However, the currently suggested optical implementation of r-PUFs has the property that all responses are altered by the reconfiguration operation, disabling it as an Erasable PUF. For electrical implementations of r-PUF based on phase-change materials, which are only briefly mentioned asides in [17], it is yet unclear whether they would be Strong PUFs at all, i.e. whether they could be designed to withstand full read-out in short time.

Eventually, there is one recent Strong PUF candidate that seems appropriate to implement Erasable PUFs: So-called Crossbar-based PUFs. They have originally been introduced in [14] [15] [16], and will be treated in the next section.

5 Strong PUFs based on Crossbar Structures

Recent work [14] [15] [16] investigated the realization of a special type of Strong PUF (so-called "SHIC PUFs" [4]). These are Strong PUFs with the additional following properties:

[4] SHIC abbreviates the term "Super-High Information Content", and is pronounced as *"chique"*.

(i) The PUF possesses maximal information content and density, with all CRPs being mutually (i.e. pairwise) information-theoretically independent.
(ii) The PUF can only be read out at slow rates.

The motivation behind investigating this type of Strong PUFs was to protect PUFs against any modeling attacks. Such attacks use known CRPs in order to extrapolate the PUF's behavior on new CRPs, and constitute a serious challenge for the security of Strong PUFs [6]. SHIC PUFs are automatically invulnerable against such modeling attempts, since all of their CRPs are information-theoretically independent: Knowing a subset of CRPs hence does not allow conclusions about other CRPs.

Concrete target parameters for the construction of SHIC PUFs discussed in [14] [15] [16] were an information content of up to 10^{10} bits and read-out speeds of 10^2 to 10^3 bits per second. As argued in [15], such relatively slow read-out speeds are no problem in many typical applications of Strong PUFs, such as bank card identification, key exchange, or also oblivious transfer [18]. On the upside, the combination of slow read out and high information content can potentially immunize the PUF against full read-out for up to month or years of uninterrupted, unnoticed adversarial access [15]. For comparison, several known Strong PUF architectures with a MHz read-out rate can be modeled (and hence broken) via a number of CRPs that can be read out in a few seconds [6].

It has been shown in [14] [15] [16] that SHIC PUFs can be realized by large, monolithic crossbar architectures. At each crosspoint of the crossbar, a diode with a random current-voltage characteristic is present. The necessary random variation in the diodes is generated by a random crystallization technique known as ALILE process. We will review the necessary basics of this approach in this section; much further detail can be found in [14] [15] [16].

ALILE Crystallization. In order to construct a Strong PUF with the above properties, one first requires a solid-state fabrication process that generates a maximal amount of entropy in the PUF. The authors of [14] [15] [16] turned to crystallization processes to this end, since the crystallization step amplifies minuscule variations in the starting conditions (such as atomic-scale roughness) to larger, stable variations in the system (for example the shape, size and position of the crystallites). Among many possible crystallization processes, they eventually selected the so-called aluminum-induced layer exchange (ALILE) process [20] [21], since it is a simple crystallization process that involves few production steps and inexpensive starting materials. It results in polycrystalline films with p-type conduction [22], and creates a highly disordered and random structure comprising of crystallized silicon grains (Si) and aluminum (Al). Fig. 3 a depicts the top view onto a crystallized system, illustrating the occurring randomness. By changing the process parameters, the size and density of the grains can be tuned as desired.

Diodes and Crossbar Read-Out. In order to read out the information contained in the system, a circuit architecture known as crossbars can be employed. It consists of two sets of parallel wires, one of them applied on the top, the other one at the bottom of the structure. Both sets are arranged orthogonally to each other. The basic schematics are

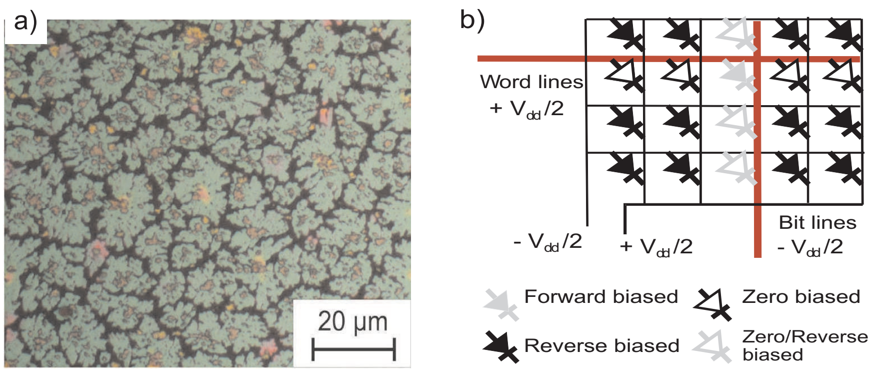


Fig. 3. (a) Randomly shaped and located Si crystallites (top view, showing the extension in x-y-directions). (b) Schematic illustration of the crossbar architecture and the diodes at the crossings. Also read-out process, i.e. the selection of a bit line and a word line in order to address and read out a single diode, is illustrated.

illustrated in Fig. 3 b. Due to the p-n-type cross section of the entire system (the film of p-type conduction is generated on an n-type wafer to this end), each virtual crossing of the crossbar acts like a p-n-diode, with rectification rates of up to 10^7 [16]. Its $I(V)$ curve can be read out by applying a voltage at two chosen crossbar wires (bit and word lines, in analogy to a memory), as illustrated in Fig. 3 b [15]. Due to the random nature of the ALILE crystallization process, the diodes show current-voltage curves which are very irregular and individual in shape. The individual curves differ in their currents by up to four decimal orders of magnitude, but are still stable against aging and multiple measurement [14] [16]. As shown in [14], at least three bits of information can be extracted reliably from each crossing.

Information Content and Inherently Slow Read-Out Speed. Using crossbar architectures has two advantages. First, they can reach ultimate information densities due to their very simple architecture of parallel wires. The information density and content targeted in [15] were 10^{10} bits per cm^2. Secondly, they can be designed with an inherently limited read-out speed. To achieve this, the Crossbar PUF is built in one large, monolithic block, not from separate blocks as modern semiconductor memories, and is made from wires that have only finite current-carrying capacity. Simulations conducted in [15] showed that in such large monolithic blocks, several milliseconds must elapse before the sense current/voltage stabilizes. This results in read-out speeds of around 100 bits/sec. Any faster read-out attempts would overload and destroy the wires, leaving the remaining structure unusable [15].

6 Erasing Information from Crossbar Structures

We now investigate if – and how – information can be erased from Crossbar PUFs. Since the information is contained in the diodes' current-voltage characteristics, any erasure operation must target the diodes, changing their $I(V)$-curves irreversibly. We could not build a device with 10^{10} crossings within the scope of this paper, but argue on the basis of measurement curves obtained from stand-alone fabricated in our group. The fact that the behavior of these single diodes scales very well to the operation of large, monolithic blocks of diodes has been proven in all detail in earlier work [15].

The "erasure operation" works as follows. A specific diode in the crossbar array is chosen by selecting the corresponding bit and word lines of the crossbar structure, similar to the read-out procedure for the crossbars. Then a short voltage pulse of 4 V to 5 V is applied in reverse direction to the diode. This induces a breakdown in the ALILE diode, which destroys the individual information present in the $I(V)$ curve, and makes all curves after erasure "standardized" and very similar in shape.

This effect has been observed by us in *all measured diodes*; three illustrative examples for $I(V)$-curves before and after breakdown are shown in Fig. 4. While the large variations in the original curves range over four orders of magnitude, there is little individuality left after breakdown. The curves after breakdown also differ strikingly from the original curves. Considering the development of the relative positions of the curves over the full voltage range shows that not even the relative positioning of the curves is preserved.

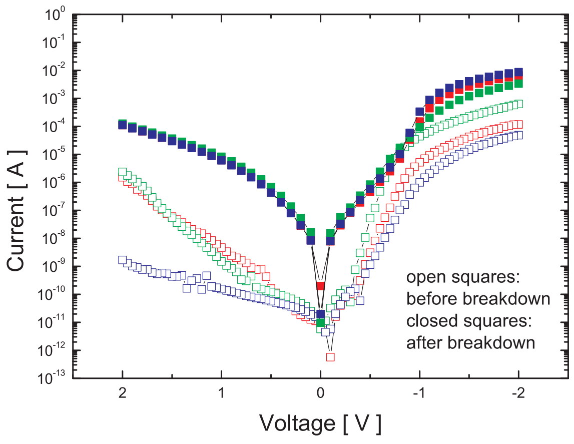


Fig. 4. The curves of three exemplary diodes (red, blue and green) before and after breakdown.

The fact that the new curves are uncorrelated to the old ones is a consequence of the physical effect behind the breakdown of the diodes. Our explanation of this mechanism is the presence of a thin natural oxide film between the p- and n-layers, effectively resulting in a p-i-n-structure. Such an additional i-layer would strongly reduce the tunneling current in reverse direction (as observed by us), which otherwise had to be expected to be high due to the large hole carrier concentration in the ALILE layers (up to 10^{19} cm^{-3}) [16]. The assumption of an intermediate oxide layer is further supported by the fact that diodes which were exposed to hydrofluoric acid (HF) vapor prior to the deposition of the ALILE layers *did not show* comparable rectification rates; the HF vapor is known to remove Si-oxide, leading to a destruction of the possible p-i-n -structure [23]. The described voltage pulse in reverse direction then simply burns and removes this i-layer.

This physical mechanism behind the erasure supports the security of our construction, for the following reasons: First, the destruction of the thin, irregular oxide film cannot be reversed physically by Eve. Second, after the oxide layer has been removed, independent and secondary features of the structure dominate the $I(V)$ curve (whereby their effect on the randomness of the curve is by far not as strong as the original configuration, see Fig. 4). From knowing the new curves after breakdown, it is therefore impossible to conclude backwards on the shape of the original $I(V)$ curves before breakdown.

Finally, please note that the operational voltage for measurement of the diodes in the crossbar structure lies between -2V and +2V. The erasure operation hence is just a factor of around 2 away from the standard operation of the crossbar. This is compatible with the use of wires with finite current-carrying capacity, which was indispensable to enforce the slow read-out rate of the crossbar (see Section 5, page 11, and [15]).

7 Summary

We made the following contributions in this paper. First, we observed a security problem in a recently published session key exchange protocol by Tuyls and Skoric [1], which is based on Strong Physical Unclonable Functions (PUFs). We cryptanalyzed the protocol under the relatively mild presumptions that the adversary gains access to the PUF twice, that she can eavesdrop the communication between the involved parties, and that no secret information can be stored on the card. As discussed earlier, these presumptions are well-motivated, for example in the bank card scenario in which the protocol had been proposed originally. Our attack has severe consequences for the security of the proposed bank card application. The noted security problem seems to be general, applying to any comparable session key exchange based on Strong PUFs.

Second, we introduced a new PUF variant, so-called Erasable PUFs, in order to resolve the described security issue. These are special Strong PUFs, with the additional property that the information stored in single responses of theirs can be irreversibly erased without changing any other response values. As we argued, currently known PUF architectures are unsuited to this end: They either are no Strong PUFs in the first place. Or, they have many interplaying components, which prevents that a single response can be changed without affecting the other responses. The latter problem holds for all delay-based PUFs, but also for the current, optical implementations of Reconfigurable PUFs.

We therefore, thirdly, investigated new architectures for implementing Erasable PUFs. We suggested the use of crossbar structures with randomly crystallized ALILE-diodes. It was known from recent work [14] [15] [16] that such "Crossbar PUFs" can act as Strong PUFs with very high information content and densities and inherently slow read-out speed. We now discussed how the information stored in the ALILE-diodes of the crossbar can be erased individually. Our erasure process works by applying a relatively small threshold current to selected bit and word lines of the crossbar. This induces a "breakdown" in the diode, as it burns intermediate oxide layers. The process is irreversible, and transforms the individual $I(V)$ curve of any diode into an uncorrelated, new one. The threshold current is low enough to be compatible with the finite current carrying capacity of the crossbar wires and the read-out mechanism of the crossbar array. We supported our proposal by measurements on single, stand alone ALILE-diodes fabricated in our group. It had been shown in extensive simulations in previous work [15] that the behavior of such diodes scales to large diode arrays.

Acknowledgements

This work was conducted in the course of the Physical Cryptography Project at the TU München, with support by the Institute for Advanced Study (IAS) and International Graduate School of Science and Engineering (IGSSE) at the TU München. We would like to thank Paolo Lugli, Martin Stutzmann, György Csaba, Ahmed Mahmoud and Michael Scholz for useful discussions.

References

1. 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.
2. R. Pappu, B. Recht, J. Taylor, N. Gershenfeld: *Physical One-Way Functions*, Science, vol. 297, pp. 2026-2030, 20 September 2002.
3. Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, Pim Tuyls: *FPGA Intrinsic PUFs and Their Use for IP Protection*. CHES 2007: 63-80
4. Blaise Gassend, *Physical Random Functions*, MSc Thesis, MIT, 2003.
5. Sandeep S. Kumar, Jorge Guajardo, Roel Maes, Geert Jan Schrijen, Pim Tuyls: *The Butterfly PUF: Protecting IP on every FPGA.* HOST 2008: 67-70
6. U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, J. Schmidhuber: *Modeling Attacks on Physical Unclonable Functions.* Accepted at ACM Conference on Computer and Communications Security, 2010. Previous versions available from Cryptology ePrint Archive, Report 251/2010, 2010.
7. U. Rührmair, H. Busch, S. Katzenbeisser: *Strong PUFs: Models, Constructions and Security Proofs.* To appear in A.-R. Sadeghi, P. Tuyls (Editors): Towards Hardware Intrinsic Security: Foundation and Practice. Springer, 2010.
8. U. Rührmair, J. Sölter, F. Sehnke: *On the Foundations of Physical Unclonable Functions.* Cryptology e-Print Archive, June 2009.
9. 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.
10. J.-W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas. *A technique to build a secret key in integrated circuits with identification and authentication applications.* In Proceedings of the IEEE VLSI Circuits Symposium, June 2004.
11. M. Majzoobi, F. Koushanfar, M. Potkonjak: *Lightweight Secure PUFs.* IC-CAD 2008: 607-673.

12. 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 - 12th International Workshop on Cellular Nonlinear Networks and their Applications, 2010.
13. Pim Tuyls, Geert Jan Schrijen, Boris Skoric, Jan van Geloven, Nynke Verhaegh, Rob Wolters *Read-Proof Hardware from Protective Coatings*. CHES 2006: 369-383
14. U. Rührmair, C. Jaeger, C. Hilgers, M. Algasinger, G. Csaba, and M. Stutzmann: *Security Applications of Diodes with Unique Current-Voltage Characteristics*. Financial Cryptography and Data Security, 2010.
15. U. Rührmair, C. Jaeger, M. Bator, M. Stutzmann, P. Lugli, and G. Csaba: *Cryptographic Applications of High-Capacity Crossbar Memories*. IEEE Transactions on Nanotechnology, 99,1, 2010.
16. C. Jaeger, M. Algasinger, U. Rührmair, G. Csaba, M. Stutzmann: *Random pn-junctions for physical cryptography.* Applied Physics Letters, Vol. 96, 172103, 2010.
17. K. Kursawe, A.-R. Sadeghi, D. Schellekens, B. Skoric, P. Tuyls: *Reconfigurable Physical Unclonable Functions – Enabling Technology for Tamper-Resistant Storage.* HOST 2009: 22-29
18. U. Rührmair: *Oblivious Transfer based on Physical Unclonable Functions (Extended Abstract)*. TRUST Workshop on Secure Hardware, Berlin (Germany), June 22, 2010. Lecture Notes in Computer Science, Volume 6101, pp. 430 - 440. Springer, 2010.
19. G. E. Suh, S. Devadas: *Physical Unclonable Functions for Device Authentication and Secret Key Generation.* DAC 2007: 9-14
20. O. Nast and S.R. Wenham: *Elucidation of the layer exchange mechanism in the formation of polycrystalline silicon by aluminum-induced crystallization.* Journal of Applied Physics, Vol. 88, pp. 124-132, 2000.
21. O. Nast and A.J. Hartmann: *Influence of interface and Al structure on layer exchange during aluminum-induced crystallization of amorphous silicon.* Journal of Applied Physics, Vol. 88, pp. 716-724, 2000.
22. T. Antesberger, C. Jaeger, M. Scholz and M. Stutzmann: *Structural and electronic properties of ultrathin polycrystalline Si layers on glass prepared by aluminum-induced layer exchange.* Appl. Phys. Lett. 2007, Vol. 91, Page 201909.
23. R. J. Carter and R. J. Nemanich: *HF vapour cleaning of oxide on c-Si*. Properties of Crystalline Silicon, EMIS Datareviews Series No 20, University of Virginia, USA, 1999.
24. G. Majni and G. Ottaviani: *Growth kinetics of (111)Si through an Al layer by solid phase epitaxy.* Journal of Crystal Growth 46, 119, 1979.
25. Daihyun Lim: *Extracting Secret Keys from Integrated Circuits*. MSc Thesis, MIT, 2004.
26. M. Majzoobi, F. Koushanfar, M. Potkonjak: *Testing Techniques for Hardware Security.* IEEE International Test Conference, 2008.

Chapter 6

PUFs in Security Protocols: Attack Models and Security Evaluations

The sixth chapter of this thesis concludes our practical security analyses of advanced PUF-protocols, and constitutes perhaps the most elaborate work of ours in this direction. Inspired by the previous chapters, and in partly generalizing on them, we introduce two formal attack models for advanced PUF protocols: The *"PUF re-use model"* and the *"bad PUF model"*. Both directly arise from practical PUF application scenarios.

In short, the PUF re-use model assumes that each PUF is used more than once in different protocols, and that therefore also adversaries may gain access to the PUF more than once, for example during different protocols executions. From a practical perspective, such an attack model seems rather inevitable, as a one-time use (and subsequent destruction) of PUFs is not very economic. Still, such one-time use scenarios have been considered and modeled in several recent theory works, including the work of Bruzska, Fischlin, Schröder and Katzenbeisser at CRYPTO 2011 [6], and another publication by Ostrovsky, Scafuro, Visconti and Wadia at EUROCRYPT 2013 [40].

The bad PUF model, on the other hand, assumes that also the manufacturer of a PUF may act maliciously, and might not fabricate a PUF in the foreseen way. Rather, he could manipulate or bias the PUF, or even replace it with an entirely different circuit with hidden properties. As electrical Strong PUFs only communicate with honest users via a digital input-output interface, such bad PUF attacks are very hard to detect for honest users. How could they verify what lies behind the digital interface, without opening and inspecting it physically?

We show in this chapter that many existing PUF-protocols for tasks like oblivious transfer or key exchange are indeed vulnerable in one of the above two attack scenarios, rendering them unusable without amendments in practical applications. We stress in this context that basic PUF schemes like Strong PUF based identification (see Chapter 8) or PUF based key storage appear generally less affected by our two attack models.

Their exact effect remains to be determined in future works.

The chapter then discusses two hardware-based countermeasures, namely so-called *"Erasable PUFs"* (see also Chapter 5) and *"Certifiable PUFs"*. Their area and cost efficient implementation would restore the general and practical usability of PUFs in advanced cryptographic protocols, and is posed as an important future, but non-trivial challenge to the PUF community.

The paper that we use in this chapter is:

- U. Rührmair, M. van Dijk: *PUFs in Security Protocols: Attack Models and Security Evaluations*. IEEE Symposium on Security and Privacy 2013, pp. 286-300, 2013.

The work has been published at the premium cyptography and security venue, the IEEE Symposium on Security and Privacy, with an acceptance rate of 12% in 2013. According to Google scholar, it has been cited 52 times to this date [26].

Again, the candidate would like to express his gratitude to have one of the founding fathers of PUFs as a co-author in this paper.

PUFs in Security Protocols: Attack Models and Security Evaluations

Ulrich Rührmair
Computer Science Department
Technische Universität München
80333 München, Germany
ruehrmair@in.tum.de

Marten van Dijk
CSAIL
MIT
Cambridge, Massachusetts
marten@mit.edu

***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). In these works, so-called "Strong PUFs" are regarded as a new, fundamental cryptographic primitive of their own, comparable to the bounded storage model, quantum cryptography, or noise-based cryptography. This paper continues this line of research, investigating the correct adversarial attack model and the actual security of such protocols.**

In its first part, we define and compare different attack models. They reach from a clean, first setting termed the *"stand-alone, good PUF model"* to stronger scenarios like the *"bad PUF model"* and the *"PUF re-use model"*. We argue why these attack models are realistic, and that existing protocols would be faced with them if used in practice. In the second part, we execute exemplary security analyses of existing schemes in the new attack models. The evaluated protocols include recent schemes from Brzuska et al. published at Crypto 2011 [1] and from Ostrovsky et al. [18]. While a number of protocols are certainly secure in their own, original attack models, the security of *none* of the considered protocols for OT, BC, or KE is maintained in all of the new, realistic scenarios.

One consequence of our work is that the design of advanced cryptographic PUF protocols needs to be strongly reconsidered. Furthermore, 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 [8] (such as Logically Erasable and Logically Reconfigurable PUFs [13]) 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, which we pose to the PUF hardware community in this work.

Keywords-**(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 [21], 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)* was 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 $r = f_P(c)$. Thereby the tuples (c, r) are usually 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 [20], [9], key exchange [20], and various forms of (tamper sensitive) key storage and applications thereof, such as intellectual property protection or read-proof memory [11], [15], [32].

In recent years, however, 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* [9] or *Strong PUF* [11], [30], [29], [23] in the literature (see also Appendix A). [1] 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.

In 2010, Rührmair [22] 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 [14], 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 [23]. They present definitions and a reductionist security proof for Strong PUF based identification. In 2011, Rührmair, Jaeger and Algasinger [27] discuss an attack on a PUF-based session key exchange scheme of Tuyls and Skoric [33], 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. Their attack motivated our PUF re-use model. At CRYPTO 2011 Brzuska, Fischlin, Schröder and Katzenbeisser [1] adapted Canetti's universal composition (UC) framework [3] to include PUFs, giving PUF-protocols for oblivious transfer (OT), bit commitment (BC), and key exchange (KE). At CHES 2012, Rührmair and van Dijk [25] 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. Two very recent eprint papers continue this general line of work: Ostrovsky, Scafuro, Visconti and Wadia [18] [2] investigate the use of so-called "malicious PUFs", and furthermore extend Brzuska et al.'s communication model in the UC framework.In independent and simultaneous work, van Dijk and Rührmair proposed a model equivalent to "malicious PUFs" under the name "bad PUF model", and a new attack model termed "PUF re-use model". The authors devise the first impossibility results for PUF-protocols in these two models [6].

While the body of work on (Strong) PUFs in cryptographic protocols is obviously growing, most papers use different implicit attack models, making it difficult to compare their results. There are situations where practically relevant attacks exist on protocols that are provably secure in other, perhaps mainly theoretical models. This motivates a comparative, systematic study of attack models.

Scope of this Work: This paper continues the above line of research. It investigates the UC-models of Brzuska et al. [1] and Ostrovsky et al. [18], and introduces several other, practically relevant attack scenarios. These include the *"stand-alone, good PUF model"*, the *"bad PUF model"*, and the *"PUF re-use model"*:

1) In the *stand-alone, good PUF model*, we assume that there is only one single, isolated protocol execution, and that all parties faithfully generate and never manipulate PUF hardware.
2) In the *PUF re-use model*, we extend this setting, and allow adversaries 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.
3) In the *bad PUF model*, we allow 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 scenario equivalent to the bad PUF model has been introduced under the name "malicious PUFs" by Ostrovsky, Scafuro, Visconti and Wadia in an eprint paper [18], their work being independent and simultaneous to the first publication of the bad PUF model by van Dijk and Rührmair in another eprint [6].

In order to illustrate the effect of the new models, we carry out exemplary security analyses of several protocols of Brzuska et al. [1] and Ostrovsky et al. [18] in the bad PUF and PUF re-use model.

Our Results: Our analyses of existing protocols show the following outcome.

1) A recent OT protocol of Brzuska et al. [1] is insecure in the PUF re-use model and in the bad PUF model. The attacks are presented in Sections III-A and III-B.
2) A recent KE-protocol of Brzuska et al. [1] is insecure in the PUF re-use model and in the combined PUF re-use, bad PUF model. The respective attacks are presented in Sections III-C and III-D.
3) A recent BC-protocol of Ostrovsky et al. [18] has certain vulnerabilities in the bad PUF model and in the combined PUF re-use, bad PUF model. This topic is discussed in Section III-E.

The above, exemplary security evaluations are carried out in full detail. In addition to that, we observe that several other

[1]We stress that the Weak/Strong PUF terminology, which was originally introduced by Guajardo, Kumar, Schrijen and Tuyls [11], is certainly not meant or to be misunderstood in a judgemental or pejorative manner.

[2]This paper has been accepted at Eurocrypt 2013 very recently, but we had only access to the eprint version [18] at the time of writing.

known PUF-protocols are insecure in the bad PUF and the PUF re-use model. Since the attacks are very similar to the abovementioned, we merely sketch them for space reasons in Section III-F. They include the following:

4) An early OT-protocol of Rührmair [22] and an early KE-protocol by van Dijk [5] are insecure in the bad PUF and the PUF re-use model (see Section III-F).
5) An OT-protocol of Ostrovsky et al. [18] is insecure in the bad PUF and the PUF re-use model (see Section III-F).
6) Two special BC-protocols of Ostrovsky et al. [18], and consequently their construction for UC-secure computation built on these two protocols, are insecure in the bad PUF model, too (see Section III-F). The attacks require the use of more complex bad PUF constructions such as Communicating PUFs and Marionette PUFs, though (see Section II-E for an explanation of the latter two).

Two important aspects should not go unnoticed. First, apart from the vulnerability in Ostrovsky et al.'s BC protocol mentioned in item 3 above, **all** of the presented attacks are outside the original attack models of the respective papers. However, we argue in great detail in Section II why the new attack scenarios must be considered realistic, and why the protocols would be faced with them in any practically relevant settings.

Secondly, our attacks in the bad PUF model require only very mild forms of bad PUFs. The attack in item 3 utilizes a bad PUF that implements a simple linear function (see Section III-E). Furthermore, the attacks of items 1, 2, 4 and 5, merely require so-called Challenge-Logging PUFs and Simulatable PUFs. The only exception is the attack mentioned in item 6: It requires a more sophisticated type of PUFs, namely Communicating PUFs (or special variants of it, such as Marionette PUFs); see Section II-E.

Besides the above new findings, two already published results should be added to complete the picture:

7) A PUF-based session key exchange protocol by Tuyls and Skoric [33] has already been attacked by Rührmair, Algasinger and Jaeger [27] under conditions similar to the PUF re-use model (without explicitly using this term). Their attack partly motivated the formal introduction of the PUF re-use model in this paper.
8) There are quadratic attacks on the security of the OT- and BC-protocol of Brzuska et al. [1] which have been presented at CHES 2012 by Rührmair and van Dijk [25]. They show that the security of these protocols is not maintained if optical PUFs or electrical PUFs with challenge length of 64 bits are used in their implementation.

As indicated by the above items (1) to (8), our analysis focuses on the impact of our attack models for "advanced" PUF protocols like OT, BC and KE. The elementary PUF use as internal key storage element and in simple identification protocols [19], [20] appears less affected (see Section V).

Consequences: 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; 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 [18]. 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.

A second option to encounter our attacks is to add two new hardware features to Strong PUFs. Firstly, one can require that Strong PUF's responses should be *"erasable"*, meaning that single responses can be "erased" (made unreadable for good). Ideally this erasure should not affect other responses; if this requirement is hard to realize in practice, then also concept similar to the logical reconfigurability of PUFs [13] may be applicable in certain settings (see Section IV). This step immunizes Strong PUF protocols against PUF re-use attacks. Secondly, Strong PUFs should be *"certifiable"*, meaning that parties holding a Strong PUF can verify that the PUF has been produced faithfully and has not been manipulated in any way afterwards. This guarantees security in the bad PUF model. The combination of both features can fully restore the applicability of Strong PUFs in concurrent, complex application environments without further restrictions (such as the above one-time use of PUFs). The implementation of these features, however, constitutes a challenging open problem that we pose to the community in this work.

Organization of this paper: In Section II we discuss and introduce various attack models for Strong PUF protocols. In Section III, we evaluate the security of several existing protocols in the new attack models. Section IV discusses the consequences of our work, in particular the need for Erasable PUFs and Certifiable PUFs. Section V summarizes the paper.

The appendix provides extra information: In Appendix A we give background on Strong PUFs to the readers who are not familiar with this concept. In Appendices B, C and D we provide some of the analyzed PUF-protocols from Brzuska et al. and Ostrovsky et al. for the convenience of the readers.

II. Attack Models for Strong PUF Protocols

Building on the general description of Strong PUF protocols in the introduction and also in Appendix A, we will now describe a number of attack scenarios for Strong PUF protocols.

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.

It seems that several early Strong PUF protocols were more or less implicitly designed for a stand-alone, good PUF setting, for example van Dijk's key exchange scheme [5] and Rührmair's OT protocol [22]. The stand-alone model will neither be realistic nor efficiently realizable in most practical PUF-applications, but makes a clean first scenario for studying the security of PUF-protocols. For practical appliances it needs to be extended, as described below.

B. The UC-Model of Brzuska et al.

In order to model the execution of multiple PUF protocols, Brzuska, Fischlin, Schröder and Katzenbeisser [1], [2] proposed one possible method how Canetti's UC-framework [3] can be adapted to PUFs. For a detailed treatment we refer the readers to the original papers [1], [2], but summarize the features of their model that are most relevant for us 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.

For completeness we indicate where the above features are specified in [2]: Features 1 and 2 directly follow from the specification of the ideal PUF-functionality $\mathcal{F}_{\mathsf{PUF}}$, in particular the first and third dotted item of Fig. 2 of [2]. Regarding feature 2, the functionality $\mathsf{init}_{\mathsf{PUF}}$ specifies that $\mathcal{F}_{\mathsf{PUF}}$ turns into the waiting state if the session `sid` already contains a PUF. And the functionality $\mathsf{handover}_{\mathsf{PUF}}$ specifies that `sid` remains unchanged in the handover, i.e., the PUF remains in the same session `sid` after the handover process. Feature 3 follows from the treatment of the subsessions `ssid` throughout their paper [2]. Examples include Figs. 3 to 8, the protocols given in Figs. 3 and 7, or the proof of Theorem 7.1, where the adversary is only allowed to access the PUF in the set-up phase, but not during or between the different subsessions.

Please note that the above features are not rudimentary aspects of the model of [1], [2], but are central to the security of their protocols and the validity of their security proofs.

C. The UC-Model of Ostrovsky et al.

Ostrovsky, Scafuro, Visconti and Wadia modify the UC-model of Brzuska et al. in a number of aspects in a recent eprint [18]. Among other things, they suggest an attack scenario termed "malicious PUFs". It is equivalent to the "bad PUF model" proposed independently by van Dijk and Rührmair [6], which is detailed in Section II-E of this paper; both models seem to have been developed independently and simultaneously.

The two author groups use their equivalent models for different purposes, though: Ostrovsky et al. give several protocols that are purportedly still secure under use of malicious/bad PUFs. Most of their constructions employ three extra assumptions: (i) they use *additional, classical computational assumptions* alongside with PUFs; (ii) they assume that the bad PUFs do not communicate with the malicious parties (compare Section II-E); and (iii) they assume that the PUFs are used only once, and can be kept away for good from the adversary or destroyed afterwards. On the other hand, van Dijk and Rührmair show that if one wants to design PUF-protocols that *solely* rest on the security of the employed PUFs, i.e., *without* additional computational assumptions, then the existence of malicious/bad PUFs leads to hard impossibility results.

We remark that in practice, the above assumption (iii) would have to be realized by destroying the PUF after each protocol, or by locking it away for good. In commercial applications, such a measure would probably be too costly and economically infeasible. The PUF re-use model in the next Section II-D investigates the consequences if it cannot be realized in practice.

D. The PUF Re-Use Model

Let us now step by step extend the model of Brzuska et al. [1], [2], and partly also of Ostrovsky et al. [18]. 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 (see Section II-B). However, this assumption seems difficult to guarantee in many natural PUF appliances.

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 [20], [19]. 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. [18] deal with this observation in their own manner: As described in Section II-C, they implicitly assume a one-time use of the PUF. Such one-time use, and subsequent destruction or locking away of the PUF, results in substantial practical costs, however. It constitutes a theoretically acceptable, but at the same time commercially somewhat infeasible measure. These considerations motivate the following attack model:

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, since it has special relevance for us.

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.

Please note that the PAM is arguably the mildest possible form of the PUF re-use model. Still, it suffices to successfully attack many existing schemes (see Section III).

E. The Bad PUF Model

One other central assumption in the UC-model of Brzuska et al. is that the players are not allowed to use "bad", fraudulent PUF-hardware with properties beyond the expected PUF functionality. This assumption can again be difficult to uphold in practice, as has been observed independently by Ostrovsky et al. [18] (see Section II-C).

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 breaks one of the essential security features of the purported "PUF" on the bankcard, namely its unpredictability. It is not too difficult to see that 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 [22] and Brzuska et al. [1] are no longer secure. If the CA acts as OT-receiver, for example, it can learn both bits of the OT-sender (see Section III-B for details).

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 [33].

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 in this paper 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.

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. [3]

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).

CL-PUFs can be implemented particularly easily in any *integrated* optical or electrical PUFs. But even for Pappu's optical, non-integrated PUF [20] challenge logging appears feasible. Imagine a special, transparent, additional layer on top of Pappu's light scattering token, which is altered by the incoming laser light. The alteration of the layer would

[3]The replacement of the internals of a PUF by a pseudo-random function is particularly hard to detect for any integrated PUFs (be they optical or electrical), since they communicate with external parties only via their integrated, digital CRP-interface; the PUF is never measured directly by the external parties. Such integrated PUFs constitute the clear majority of currently investigated PUFs. But even for Pappu's optical PUF, simulatability can be an issue: It is by no means ruled out that the adversary builds a light scattering token that has a particular, well-ordered structure, which leads to simple and simulatable outputs. Current protocols would not even detect if the adversary used an "empty" plastic token, which did not contain any scatterers at all, and which was trivially simulatable.

not necessarily be visible by the sheer eye, but could reveal itself only under UV-light or other special illumination. Such a sensitive layer would indicate the point of incidence (and perhaps even the angle) of the challenge, i.e., it would show some form challenge logging.

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.

Discussion of Potential Countermeasures: 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. Physical inspection of the inner configuration of the PUF as a regular protocol step seems no viable possibility, as discussed in the previous paragraphs. 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. Once more, this makes bad PUFs a realistic and efficient method to cheat.

Brzuska et al. [1] indeed assume certification of the PUF, but do not give protocols or methods how it can be achieved. For the above reasons, we believe that efficient certification is currently infeasible in practice. This holds even more if malicious players, and not only external adversaries, generate and use manipulated PUFs. We comment that in a typical two-party protocol, a PUF originating from a malicious party must be considered as nothing else than an untrusted piece of hardware that stems from the adversary.

Advanced Bad PUFs: How "bad" can a PUF be? Having focused on simple features in the last section (which still suffice to attack many existing protocols), we will play with a number of more sophisticated properties now. The purpose of our discussion is to complement the picture; we will not fully work out every construction in detail.

To start with, it is of course possible to imagine bad PUFs that communicate information (e.g., wirelessly) to the malicious party. Such a *"Communicating PUF"* could transmit 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 were shielding the PUF during the protocol and destroying them immediately afterwards.

Another advanced bad PUF example is a PUF which transmits all challenges to the malicious party in real-time; waits for the malicious party to individually select and return a response R_{bad}; and then outputs R_{bad} (as if it was the natural response of the PUF itself). The latter type of PUF could be called the *Marionette PUF* for obvious reasons. It seems clear that there is no security benefit of using PUFs in cryptographic protocols if the adversary can use Marionette PUFs. Their employment makes PUFs useless, in the sense that for any protocol that uses PUFs and which securely implements a task T even if Marionette PUFs are employed, there will be a protocol that securely implements T and does not use the (Marionette) PUFs at all. Therefore the existence and use of Marionette PUFs must be ruled out in most advanced Strong PUF protocols such as OT, BC and KE by whatever means. One potential, but again costly countermeasure to prevent Marionette PUFs would be the shielding of the PUF during the entire course of the protocol.

A third example are bad PUFs that adapt or alter their response behavior over time. This adaption could be a function of the challenge that is applied to them, or a function of all previous challenges. Other variants of adaptive bad PUF behavior include the following: (i) The PUF could automatically alter its response behavior after a certain time period t_0. This means that the malicious party can influence the protocol by delaying the protocol; note that this is explicitly allowed in the UC-model. (ii) The PUF could change its CRPs upon a wireless triggering signal it receives. (iii) The PUF could even change upon a certain, triggering *challenge* that is applied to it. This allowed the malicious party to influence the bad PUF even while it is not in her possession, simply by causing the honest party to apply a certain challenge to the PUF.

A final example are bad PUFs that implement arbitrary digital functions f with special, fraudulent properties. Simulatable PUFs (where f is simulatable and the simulation code is known to the malicious party) are one special case of this approach. But the function f could have other handy properties for the adversary. For example, it might be a function for which the computation of inverses is simple. This case is actually relevant for our attack in Section III-E.

Many other examples of advanced bad PUFs are conceivable. Actually, any such bad PUF types have to be taken into consideration when the security of a PUF protocol is analyzed. But since the earlier, simpler types of SIM-PUFs and CL-PUFs already suffice for attacking many protocols, we will not deal too much with advanced bad PUFs further in this paper.

A Final Thought on Bad PUFs: Let us conclude this section by a general thought. Why are bad PUFs so powerful? Consider the following line of thought: Suppose that a PUF-protocol utilizes some property $\mathcal{P}$ of the employed PUF to achieve its security. Then there will (almost with certainty) be a bad PUF which is hard to recognize from the outside, but which *does not* possess the property $\mathcal{P}$. The security of the protocol and the validity of the proof will no longer be guaranteed if the adversary uses this bad PUF not possessing $\mathcal{P}$. This makes bad PUF a broadly applicable method of cheating. The cost of implementing the imagined bad PUF type determines how practically relevant the resulting attack is; we focused on relatively easily implementable variants of bad PUFs in this paper.

III. Security Evaluations in the PUF Re-Use and Bad PUF Model

We will now conduct three detailed, exemplary security analyses in the new attack models. We selected the PUF-based OT- and KE-protocol of Brzuska et al. from Crypto 2011 [1] and the recent BC-protocol by Ostrovsky et al. [18] to this end. The protocols are given in a simplified form in Appendices B, C and D for the convenience of the readers. The notation employed in our attacks actually refers to these appendices. We would like to stress that the first two protocol by Brzuska et al. are secure in their own, original attack model (apart from a recent attack on Brzuska's OT-Protocol by Rührmair and van Dijk [25]). But, as argued earlier, the protocols would likely be faced with the PUF re-use model and the bad PUF model once they were used in practice. In opposition to this, the BC-protocol of Ostrovsky et al. is actually attacked in their own, original "malicious" PUF model.

A. OT-Protocol of Brzuska et al. in the PUF Re-Use Model

We start by analyzing the OT-Protocol of Bruzska et al. [1] (see Protocol 1 in Appendix B) in the PUF re-use model, or, to be more precise, in the mildest form of the PUF re-use model, the PAM. Our attack rests on the following *assumptions*:

1) After the initialization phase of the OT-Protocol 1, different subsessions of the protocol are run. We assume that there is a subsession `ssid` with the following properties:
   - Eve was able to eavesdrop the binary communication between the sender and the receiver in the subsession `ssid`.
   - Eve can read-out CRPs from the PUF after the end of the subsession `ssid`, for example before a new subsession `ssid′` is started. (Note that this assumption is derived from the PAM.)

Under these provisions, Eve can learn both bits s_0 and s_1 used by the sender in subsession `ssid`. This breaks the security of this subsession. The attack works as follows:

1) When the subsession `ssid` is run, Eve eavesdrops the messages in Steps 3, 4 and 6. She therefore learns the values $x_0, x_1, v\ (:= c \oplus x_b), S_0\ (:= s_0 \oplus r_0)$ and $S_1\ (:= s_1 \oplus r_1)$. Thereby r_0 and r_1 are the responses to the challenges $c_0 (:= v \oplus x_0)$ and $c_1 (:= v \oplus x_1)$.
2) When Eve has got physical access to the PUF after the subsession `ssid`, she computes the challenges $c_0 := v \oplus x_0$ and $c_1 := v \oplus x_1$ herself. She applies these challenges to the PUF, and obtains the responses r_0 and r_1.
3) Eve derives s_0 and s_1 by computing the values $S_0 \oplus r_0 = s_0 \oplus r_0 \oplus r_0 = s_0$ and $S_1 \oplus r_1 = s_1 \oplus r_1 \oplus r_1 = s_1$.

This breaks the security of the subsession `ssid`.

Please note that the role of Eve can also be played by a malicous receiver. Interestingly, an attacker cannot learn the receiver's choice bit b by a similar attack, since the secrecy of the choice bit is unconditional and does not rest on the employed PUF.

B. OT-Protocol of Brzuska et al. in the Bad PUF Model

Let us now describe an attack on the OT-Protocol of Brzuska et al. [1] (see Protocol 1 in Appendix B) in the bad PUF model, which works under the following single assumption:

1) The receiver can hand over a simulatable bad PUF instead of a normal PUF in the initialization phase, and furthermore possesses a simulation algorithm for this PUF.

The attack itself works as follows:

1) The receiver follows Protocol 1 as specified, and carries out a subsession `sid`.
2) When the subsession is completed, the receiver computes the two challenges $c_0 := v \oplus x_0$ and $c_1 := v \oplus x_1$. He can do so since he knows v, x_0 and x_1 from earlier protocol steps.

3) The receiver uses his simulation algorithm in order to compute the two responses r_0 and r_1 which correspond to the challenges c_0 and c_1.
4) The receiver derives both values s_0 and s_1 by computing $S_0 \oplus r_0 = s_0 \oplus r_0 \oplus r_0 = s_0$ and $S_1 \oplus r_1 = s_1 \oplus r_1 \oplus r_1 = s_1$. He can do so since he knows S_0, S_1 from step 6 of the OT-protocol.

The Sender hence learns both strings s_0 and s_1, breaking the security of the protocol. We comment that the attack only requires the use of simulatable PUFs by the receiver, which are particularly easy to implement.

C. KE-Protocol of Brzuska et al. in the PUF Re-Use Model

We describe below how the KE-Protocol of Brzuska et al. [1] (see Protocol 2 in Appendix C) can be attacked in the PUF re-use model, or more, precisely, in its mildest form, the PAM. The attack is quite straightforward and rests on the following assumptions:

1) After the initialization phase of Protocol 2, different subsessions of the protocol are run. We assume that there is a subsession ssid with the following properties:
   - Eve was able to eavesdrop the binary communication between the Alice and the Bob in the subsession ssid.
   - Eve can read-out CRPs from the PUF after the end of the subsession ssid, for example before a new subsession ssid$'$ is started.

Under these provisions, Eve can learn the exchanged key K. The attack is relatively obvious and works as follows:

1) When the subsession ssid is run, Eve eavesdrops step 2 and learns the values c and d.
2) When Eve has got physical access to the PUF after subsession ssid, she applies the challenge c to the PUF, measures the (noisy) response r', and derives the secret st from r' by the help of d.

As $st = K$ in subsession ssid, this breaks the security of this subsession.

D. KE-Protocol of Brzuska et al. in the Combined PUF Re-Use and Bad PUF Model

Let us continue examining the security of the KE-Protocol of Brzuska et al. [1] (Protocol 2 in Appendix C). Since in a simple stand-alone scenario neither Alice nor Bob have an incentive to use bad PUFs, this is a welcome opportunity to illustrate the impact of a combined attack model: namely a combination of the PUF re-use and the bad PUF model.

We make the following assumptions:

1) The KE protocol is executed between Alice and Bob (including an initialization phase and an arbitrary number of subsessions), and later between Bob and Claire (again including an initialization phase *with the same PUF* and later subsessions).
2) Alice plays maliciously, and uses a challenge-logging PUF in her initialization phase.

Under this assumption, Alice can learn the key exchanged by Bob and Claire as follows:

1) When the PUF is in transition from Bob to Claire in step 3 of the initialization phase of their protocol, Alice gains physical access to the PUF.
2) Alice reads out the last previously applied challenge c, applies it to the PUF, and obtains response r.
3) In the next subsession phase, Alice intercepts the helper data d that is sent from Bob to Claire in step 2.
4) Alice utilizes her knowledge of r and d to infer $st = K$.

Alice hence learns the key K exchanged by Bob and Claire, breaking the protocol. Let us mention a few simple modifications of the attack: Alice could alternatively use a simulatable PUF (instead of a CL-PUF), leading to a similar attack. If the PUF is obtained by Alice from a third party manufacturer, then the manufacturer can mount the same attack by using CL- or simulatable PUFs. Finally, if an external adversary Eve is able to add a challenge logger while the PUF is in transit from Alice to Bob, then she can derive both Alice's and Bob's key as well as Bob's and Claire's key by reading out the challenge logger when the PUF is in transit from Bob to Claire. The details of these variants are similar to the attack above and are left to the reader.

E. An Unconditional BC-Protocol of Ostrovsky et al. in the Bad PUF Model

Ostrovsky et al. [18] describe in Fig. 6 of their paper (see Protocol 3 in Appendix D) an unconditional BC-protocol (i.e., one that does not use any additional computational assumptions) which is designed to be secure under the use of malicious/bad PUFs. It uses a PUF that maps inputs of length n to outputs of length $3n$.

We present an attack which works in the original communication scenario of Ostrovsky et al. [18], where the correct lengths of the strings y and q, representing the output/response and input/challenge, are not explicitly verified. Including such a check in the protocol is an essential step for achieving security in the bad PUF model. Likely such a check was implicitly assumed by Ostrovsky et al. in their protocol without making this explicit.

We describe below a relatively simple bad PUF type that allows cheating in the protocol where the input/challenge to output/response message expansion is not verified. We use a bad PUF which (i) has equal input/output lengths (instead of being length expanding), (ii) implements a permutation on the challenge space, and (iii) has outputs that are computationally easy to invert. Our attack then makes the following single assumption:

1) The committer $\mathsf{C}_{\mathsf{uncon}}$ uses/initializes a bad PUF in the protocol instead of a good PUF. For some fixed value $X \in \{0,1\}^{3n}$, this bad PUF implements a linear permutation $f : \{0,1\}^{3n} \rightarrow \{0,1\}^{3n}$, mapping each challenge C to the response $R := C \oplus X$.

The attack subsequently proceeds as follows:

1) The committer $\mathsf{C}_{\mathsf{uncon}}$ initializes the above bad PUF in the beginning of the protocol.
2) The committer can then cheat in the decommitment phase as follows: In order to deliberately open the commitment to bit $b = 0$, he sends the challenge $st \oplus X$. In order to open it to bit $b = 1$, he sends the challenge $r \oplus st \oplus X$. In the first case, the (bad) PUF outputs $st \oplus X \oplus X = st$, meaning that the receiver accepts the decommitment for $b = 0$. In the second case, the (bad) PUF outputs $r \oplus st \oplus X \oplus X = r \oplus st$, implying that the receiver accepts the decommitment for $b = 1$.

It is not too difficult to see that other attacks exist in the bad PUF model or in the combined PUF re-use, bad PUF model, i.e., if one leaves the original communication scenario of Ostrovsky et al. For example, the protocol is vulnerable if a bad PUF that communicates with the committer or the receiver is used. The receiver can use this communication channel to learn the committed bit previous to the reveal phase; and the committer can use the channel to alter the behavior of the PUF and open the commitment to his like. The same holds if a malicious receiver has equipped the PUF with a challenge-logger at an earlier occasion; this allows him to learn the committed bit previous to the reveal phase. Given our other discussions in this section, the details here are relatively straightforward and are omitted for space reasons. We stress again that while the three latter attacks go beyond Ostrovsk et al.'s original communication scenario, they still represent practically relevant strategies in our opinion (see Sections II-C to II-E). Further aspects will be discussed an extended version of this paper [26].

F. Security of Other Protocols in the PUF Re-Use Model and Bad PUF Model

For reasons of brevity, we focused on the three above protocols in our detailed security analysis. Other Strong PUF protocols for OT, BC or KE can be attacked in similar manners. Since the attacks are analog to the work in the above sections, we merely sketch and summarize them in the following list:

1) The OT-protocol of Rührmair [22] is no longer secure in the bad PUF and PUF re-use model, and similar considerations hold for the key exchange protocol of van Dijk [5]. This can be seen relatively easily, since the attacks are essentially equivalent to the attacks in the last subsections on Brzuska et al.'s OT and KE protocol.
2) It is not too difficult to see that the unconditional OT-protocol of Ostrovsky et al. for honest PUFs (see Fig. 7 of [18]) is not secure in the PUF re-use model. If the receiver of the protocol gets access to the used PUFs after the end of the protocol, he can learn both strings s^0 and s^1.
   Furthermore, if the sender uses bad, challenge-logging PUFs instead of honest PUFs as $\mathsf{sid}_1^{\mathsf{S}}, \ldots, \mathsf{sid}_{2k}^{\mathsf{S}}$, then he can obviously learn the value of the b_i, which allows him to derive the the choice bit b from the values b'_{i_j} which he receives in step 4 of the protocol. Actually, even something weaker suffices: If only one of the PUFs $\mathsf{sid}_j^{\mathsf{S}}$ for $j \in S$ is challenge logging, then b is revealed. Since $S \subset [2k]$ is a randomly chosen subset of size k, the latter condition can be enforced by merely making $k+1$ of the $2k$ PUFs challenge logging. In other words, the attack also works if only a strict subset of all PUFs are bad.
3) The statistically hiding, straight-line extractable bit commitment scheme of Fig. 10 of Ostrovsky et al. [18], and the statistically binding, straight-line extractable equivocal commitment scheme of Fig. 11 of the same paper, can be attacked by communicating bad PUFs, which maliciously transfer the challenges applied to the PUF in the commit phase to the receiver. This allows the receiver to learn the committed bit before the reveal phase.
   We stress once more that Ostrovsky et al. seem to implicitly assume that there is no communication between the malicious party and the PUF, i.e., we are again extending the original attack model of Ostrovsky et al. here. However, as discussed earlier, Communicating PUFs seem hard to prevent in certain settings. If they are considered realistic, then also the construction for UC-secure computation of Ostrovsky et al., which is built on the commitment schemes in Figs. 10 and 11 of [18], breaks down.

G. Summary of Our Security Discussion

To summarize, all Strong PUF protocols for OT, BC and KE examined in this paper can be attacked in variants of the PUF re-use model, the bad PUF model, or the combined PUF re-use, bad PUF model. Only one of these attacks (see item 3 of Section III-F above) requires Communicating PUFs, which are somewhat complex. The majority of attacks, however, can be carried out in simple variants of the bad PUF model, using simulatable or challenge-logging PUFs, or straight away in the ordinary PUF re-use model.

We stress again that most of the attacks work outside the attack scenarios and communication models of the original papers, but we argued in Section II why we consider the new models realistic. One notable exception is the attack on Ostrovsky's unconditional bit commitment scheme in the malicous PUF model (see Section III-E), which actually works in the original attack model of Ostrovsky et al.

The authors of this paper are not aware of any PUF protocols for OT, BC or KE which can withstand all said attack models, and in which (i) plain Strong PUFs with no additional hardware properties are used, (ii) no additional assumptions (set-up assumptions, classical computational assumptions, etc.) apart from the security (i.e., unpredictability) of the Strong PUF are made. This illustrates the acuteness of re-thinking current PUF protocol design.

IV. Consequences, Or: The Need for Erasable and Certifiable PUFs

What are the consequences of the observations of the last sections? The first and foremost implication is that attack models for PUF protocols should be reconsidered. PUFs are different from other cryptographic primitives in that they are real pieces of hardware that can have all kinds of malicious properties. Future protocol design and security analyses must take this into account.

One potential route to evade some of our attacks has been considered by Ostrovsky et al. in [18]. They combine three steps to construct secure PUF-protocols in the presence of malicious/bad PUFs: (i) They allow additional, standard computional assumptions in the protocols. (ii) They assume that the PUF cannot communicate with the malicious party, in particular, that the PUF is no Marionette PUF and no Communicating PUF. (iii) They assume a strict one-time use of the PUF; potentially malicious parties must be kept away from the PUF after it has been used. Measures (ii) and (iii) essentially must be realized by effectively shielding the PUF continuously until it is destroyed at the end of its one-time use. These are certainly very costly and non-trivial measures. They lead us to the question whether other approaches for fighting the PUF re-use model and bad PUFs exist in practice.

Erasable and Certifiable PUFs: Two other, direct countermeasures against the PUF re-use model and bad PUFs are so-called Erasable and Certifiable 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 [27], 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 [31] and optical PUFs [20]. 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 [16] 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 [8]. 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. [13] 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.

Certifiable PUFs, on the other hand, 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. We argued already in Section II-E why it is important that such a certification can be carried out offline: Communication with a trusted authority upon every protocol execution (in order to certify the PUF) makes the use of PUFs obsolete. One could then implement the desired functionalities easier by using the trusted authority itself. 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. 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.

V. Summary and Future Work

We introduced a number of new attack models for Strong PUF protocols in this paper, including the *"PUF re-use model"* and the *"bad PUF model"*. These models, so we argued, constitute practically relevant and hard-to-detect attack strategies, and are strongly relevant for practical PUF usage scenarios.

We then illustrated the power of the new models by analyzing the security of several known protocols. The results were already summarized in detail in Section I. In short, all analyzed oblivious transfer (OT), bit commitment (BC) and key exchange (KE) protocols for Strong PUFs can be attacked successfully in the bad PUF model and/or the PUF re-use model. This includes schemes by Rührmair [22], van Dijk [5], Brzuska et al. presented at Crypto 2011 [1], and Ostrovsky et al. [18]. With one exception, where so-called Communicating PUFs are required, all attacks in the bad PUF model only utilize very simple types of bad PUFs, such as simulatable PUFs and challenge-logging PUFs. The attacks in the ordinary PUF re-use model are even simpler to execute, and do not require physical modification of PUFs at all.

We remark **once more** that our attacks leave the original attack models of the protocols (with the single exception of the vulnerability of Section III-E). Still, our attack models seem realistic, and indeed closely follow practical usage scenarios of PUFs. Depending on the exact application, the protocols would likely be faced with them once they were used in practice. This implies that current attack models and design strategies for advanced PUF protocols such as OT, BC or KE must strongly be re-thought.

Two potential classes of countermeasures against our attacks were analyzed in Section IV: The first is the employment of classical computational assumptions in combination with a strict one-time use of PUFs and shielding of the PUFs against communication with the malicious party until its destruction [18]. This step maintains the usability of standard Strong PUFs in advanced settings and in the presence of bad PUFs, but is economically very costly and also difficult to realize in practice. Furthermore, the combination of PUFs and classical computational assumptions takes away some of the appeal of PUFs as a new, independent, and post-quantum cryptographic primitive that enables advanced protocols.

A second possibility, that would restore the usability of PUFs in complex application settings without any restrictions, is the use of Certifiable and Erasable PUFs. These are PUFs which can be certified offline for their genuineness, and for the fact that they have no other features than those expected by the honest parties ("certifiability"); and PUFs that allow the selective erasure of single PUF responses without affecting other responses ("erasability"). Without presenting a formal proof of this claim in this paper, they seem to allow efficient and secure PUF protocols whose security is built on the unpredictability of the PUF alone, without requiring additional computational assumptions. These novel PUF types could maintain the status of Strong PUFs as a general, new cryptographic primitive. If Erasable PUFs maintain hard to realize in practice, then also variants such as logical erasability/reconfigurability [13] could be interesting in our context. In order to fight both the bad PUF and the PUF re-use model, however, erasability (or variants of it) and certifiability have to be combined in a single piece of hardware. No strategies for this exist in the current literature.

Relation to PUF-Based Key Storage and Strong PUF-based Identification: Apart from their use in basic cryptographic protocols, a second established application of PUFs is their usage as (tamper-sensitive) key storage element. This application has at times been termed a "physically obfuscated key" or POK [7], sometimes also a "Weak PUF" [11]. We stress that this application scenario is not the topic of our paper. Independent of whether such an assumption is considered realistic or not, POKs explicitly suppose that the PUF's responses remain internal forever, and can only be accessed by the system itself to derive an internal secret key. This makes this PUF-type unusable for the type of protocols considered in this paper; and at the same time, it makes the attacks in the PUF re-use model meaningless. Also the bad PUF model seems obsolete: PUF-based key storage assumes that the manufacturer in a secure set-up phase can read out the key derived from the PUF, and uses it later on in communication with the PUF. This presupposes some basic trust in the manufacturer in the first place, since the secret key is shared with him from the beginning. The exact relation between POKs and the bad PUF model will be the topic of future analysis.

Something similar holds for the common Strong PUF based identification protocol by Pappu et al. [20], [19]. The use of bad PUFs here appears less relevant, and the PUF re-use model does not seem to pose a significant threat. On the other hand, a manufacturer who uses a simulatable PUF can later impersonate the PUF-carrying hardware. The exact implications of our attack models on PUF-based identification were not the topic of this paper, and are left for future investigations.

Future Work: We expect two strands of substantially new research to emerge from our findings. The first will be concerned with the theory behind Strong PUF protocols: New attack models and security definitions must be developed, for example in the context of the UC-framework. They could include the formal definition of Erasable PUFs (and variants such as Logically Erasable/Reconfigurable PUFs) and Certifiable PUFs, and the investigation of "PUF attestation" as standard protocol step. New security proofs will need to be led in these environments. Finally, the exact implications of our attack models for other PUF applications than OT, BC and KE must be determined.

The second strand of research regards PUF hardware, and concerns the development of efficient Erasable and Certifiable PUFs. As briefly addressed in Section IV, combining these two features in a single piece of hardware seems highly non-trivial. The same holds for combinations of logical erasability/reconfigurability and certifiability. We would like to pose these problems as central future challenges to the PUF hardware community in this work.

Acknowledgements

The authors would like to thank Jürg Wullschleger for his contributions on the bad PUF model and on challenge-logging PUFs, and Marc Fischlin for his comments on our attack on Ostrovsky et al.'s bit commitment protocol presented in Section III-E.

References

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

[2] C. Brzuska, M. Fischlin, H. Schröder, S. Katzenbeisser: *Physical Unclonable Functions in the Universal Composition Framework.* Full version of the paper. Cryptology ePrint Archive, Report 2011/681, 2011. Downloaded March 2012.

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

[4] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, U. Rührmair: *The Bistable Ring PUF: A new architecture for strong Physical Unclonable Functions*. HOST 2011: 134-141

[5] M. van Dijk: *System and method of reliable forward secret key sharing with physical random functions.* US Patent No. 7,653,197, October 2004.

[6] M. van Dijk, U. Rührmair: *Physical Unclonable Functions in Cryptographic Protocols: Security Proofs and Impossibility Results.* Cryptology ePrint Archive, Report 2012/228, 2012. Downloaded April 2012.

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

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

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

[10] B. Gassend, D. Lim, D. Clarke, M. van Dijk, S. Devadas: *Identification and authentication of integrated circuits.* Concurrency and Computation: Practice & Experience, 2004.

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

[12] D. E. Holcomb, W. P. Burleson, K. Fu: *Initial SRAM state as a fingerprint and source of true random numbers for RFID tags.* RFID Security, 2007.

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

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

[15] S. S. Kumar, J. Guajardo, R. Maes, G. J. Schrijen, P. Tuyls: *The Butterfly PUF: Protecting IP on every FPGA*. HOST 2008: 67-70

[16] K. Kursawe, A. R. Sadeghi, D. Schellekens, P. Tuyls, B. Skoric: *Reconfigurable physical unclonable functions – Enabling technology for tamper-resistant storage*. HOST 2009: 22-29.

[17] J.-W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas. *A technique to build a secret key in integrated circuits with identification and authentication applications.* In Proceedings of the IEEE VLSI Circuits Symposium, 2004.

[18] R. Ostrovsky, A. Scafuro, I. Visconti, A. Wadia: *Universally Composable Secure Computation with (Malicious) Physically Uncloneable Functions*. Cryptology ePrint Archive, Report 2012/143, 2012. First version downloaded in April 2012. Throughout our paper, we refer to the numbering of figures and protocols of the latest version of Ostrovsky et al. that was available at the time of preparing our camera ready paper. This latest version stems from Nov. 14, 2012.

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

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

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

[22] U. Rührmair: *Oblivious Transfer based on Physical Unclonable Functions*. TRUST 2010, pp. 430 - 440, Springer 2010.

[23] U. Rührmair, H. Busch, 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.

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

[25] U. Rührmair, M. van Dijk: *Practical Security Analysis of PUF-based Two-Player Protocols*. Cryptographic Hardware and Embedded Systems (CHES 2012), Springer, 2012.

[26] U. Rührmair, M. van Dijk: *PUFs in Security Protocols: Attack Models and Security Evaluations*. Cryptology ePrint Archive, 2013. To be submitted.

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

[28] U. Rührmair, C. Jaeger, M. Bator, M. Stutzmann, P. Lugli, G. Csaba: *Applications of High-Capacity Crossbar Memories in Cryptography*. IEEE Transactions on Nanotechnology, 2011.

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

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

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

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

[33] 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.

APPENDIX

A. Strong PUFs

Different subtypes of PUFs exist (see [29], [30], [24]), each with their own security properties and applications. Strong PUFs are an important and central of these subtypes. They have also been called Physical Random Functions due to their similarity with the more classical Pseudo-Random Functions [10]. A Strong PUF is a PUF with the following features (for formal definitions see [30], [23], [1]):

1) *Public CRP interface:* Its challenge-response mechanism is publicly accessible. Everyone who holds a Strong PUF can apply challenges to it and read out the corresponding responses.
2) *Large CRP set:* It has a very large number of challenges, ideally exponentially many in some system parameter, such as the system's physical size or the challenge length. Together with the finite read-out speed of the Strong PUF, the large number of challenges makes it impossible to read out all CRPs in a limited time, such as days or even weeks.
3) *Unpredictability:* The CRP-behavior of Strong PUFs is so complex that it cannot be modeled or machine learned or otherwise predicted. An adversary who knows a large subset of all CRPs nevertheless cannot build a model that allows him to correctly predict the response to a randomly chosen, previously unknown challenge with high probability.

The above features imply that only the very party who currently holds possession of a Strong PUF can determine the correct response to a randomly chosen challenge with high probability, even if the PUF has been in the possession of other parties before. This observation can be exploited cryptographically in various ways, as we will see later in this paper. Typical examples of Strong PUFs are given in [19], [20], [9], [31], [28], [4]. Modeling attacks on Strong PUFs have been reported, among other places, in [29].

One advantage of Strong PUFs over other types of PUFs (such as Weak PUFs/POKs, see again [29]) is that their responses do not need to remain secret, and do not require protection inside the embedding hardware.

B. OT-Protocol of Brzuska et al.

The OT protocol of Brzuska et al. [1] implements one-out-of-two string oblivious transfer. It is assumed that in each subsession the sender P_i initially holds two (fresh) bitstrings $s_0, s_1 \in \{0,1\}^\lambda$, and that the receiver P_j holds a (fresh) choice bit b.

Brzuska et al. generally assume in their treatment that after error correction and the application of fuzzy extractors, a PUF can be modeled as a function $\mathsf{PUF}: \{0,1\}^\lambda \to \{0,1\}^{rg(\lambda)}$. We often use this model throughout this paper, too. In the upcoming protocol, they furthermore assume that $rg(\lambda) = \lambda$, i.e., that the PUF implements a function $\mathsf{PUF}: \{0,1\}^\lambda \to \{0,1\}^\lambda$ (compare [1], [2]).

Protocol 1: PUF-BASED OT BY BRZUSKA ET AL. ([1], SIMPLIFIED DESCRIPTION)

External Parameters: The protocol has a number of external parameters, including the security parameter λ, the session identifier `sid`, a number N that specifies how many subsessions are allowed, and a pre-specified PUF-family $\mathcal{P}$, from which all PUFs used in the protocol must be drawn.

Initialization Phase: Execute once with fixed session identifier `sid`:

1) The receiver holds a PUF which has been drawn from the family $\mathcal{P}$.
2) The receiver measures l randomly chosen CRPs $c_1, r_1, \ldots, c_l, r_l$ from the PUF, and puts them in a list $\mathcal{L} := (c_1, r_1, \ldots, c_l, r_l)$.
3) The receiver sends the PUF to the sender.

Subsession Phase: Repeat at most N times with fresh subsession identifier `ssid`:

1) The sender's input are two strings $s_0, s_1 \in \{0,1\}^\lambda$, and the receiver's input is a bit $b \in \{0,1\}$.
2) The receiver chooses a CRP (c, r) from the list $\mathcal{L}$ at random.
3) The sender chooses two random bitstrings $x_0, x_1 \in \{0,1\}^\lambda$ and sends x_0, x_1 to the receiver.
4) The receiver returns the value $v := c \oplus x_b$ to the sender.
5) The sender measures the responses r_0 and r_1 of the PUF that correspond to the challenges $c_0 := v \oplus x_0$ and $c_1 := v \oplus x_1$.
6) The sender sets the values $S_0 := s_0 \oplus r_0$ and $S_1 := s_1 \oplus r_1$, and sends S_0, S_1 to the receiver.

7) The receiver recovers the string s_b that depends on his choice bit b as $s_b = S_b \oplus r$. He erases the pair (c, r) from the list $\mathcal{L}$.

Comments: The protocol implicitly assumes that the sender and receiver can interrogate the PUF whenever they have access to it, i.e., that the PUF's challenge-response interface is publicly accessible and not protected. This implies that the employed PUF must possess a large number of CRPs. Using a PUF with just a few challenges does not make sense: The receiver could then create a full look-up table for all CRPs of such a PUF before sending it away in Step 3 of the Initialization Phase. This would subsequently allow him to recover both strings s_0 and s_1 in Step 6 of the protocol subsession, as he could obtain r_0 and r_1 from his look-up table. Similar observations hold for the upcoming protocols: Indeed, all protocols discussed in this paper do require PUFs with a large number of challenges, a publicly accessible challenge-response interfaces, and an unpredictable CRP-behavior (or, in other words, *Strong PUFs*).

Further, please note that no physical transfer of the PUF and no adversarial access is envisaged during the subsessions of the protocol, as already indicated in Section II-B.

C. KE-Protocol of Brzuska et al.

Together with CRP-based identification, key exchange (KE) was among the first security applications suggested for PUFs. Pappu et al. were the first to mention *"key establishment"* as a potential PUF application [20], and van Dijk gives the first concrete protocol in a patent writing [5]. The KE protocol of Brzuska et al. [1] picks up these known approaches. We again describe it in a simplified form.

Protocol 2: PUF-BASED KEY EXCHANGE ([1], SIMPLIFIED DESCRIPTION)

External Parameters: The protocol has a number of external parameters, including the security parameter λ, the session identifier `sid`, a number N that specifies how many subsessions are allowed, and a pre-specified PUF-family $\mathcal{P}$, from which all PUFs used in the protocol must be drawn.

Initialization Phase: Execute once with fixed session identifier `sid`:

1) Alice holds a PUF which has been drawn from the family $\mathcal{P}$.
2) Repeat N times:
   - Choose a challenge c at random, measure the response r of the PUF, create helper data d, and extract a secret st from r. Add the tuple (c, r, st, d) to the list $\mathcal{L}$.
3) Alice sends the PUF to Bob.

Subsession Phase: Repeat at most N times with fresh subsession identifier `ssid`:

1) Alice picks a tuple (c, r, st, d) from the list $\mathcal{L}$ at random.
2) Alice sends (c, d) to Bob over the authenticated binary channel.
3) Bob measures a (possibly noisy) response r' to the challenge c. He uses the helper data d to recover the same secret st as the Server.
4) Both Alice and Bob set their key $K = st$. Alice erases the tuple (c, r, st, d) from the list $\mathcal{L}$.

Comments: For the same reasons as discussed in Section B, the above KE protocols assumes (and indeed requires) a Strong PUF. If the PUF has only got a small CRP-set, then the adversary can fully read out all CRPs when the PUF is in transition from Alice to Bob. Furthermore, no adversarial access is foreseen or allowed between the different subsessions of the protocol.

D. An Unconditional BC-Protocol of Ostrovsky et al. in the Malicious/Bad PUF Model

Ostrovsky et al. [18] give an unconditional BC-protocol (i.e., one that does *not* use computational assumptions) in Fig. 6 of their paper. The protocol is intended to be secure in the malicious PUF model. The protocol assumes a PUF with challenge length n and response length $l = 3n$.

Protocol 3: PUF-BASED BC IN THE MALICIOUS PUF MODEL BY OSTROVSKY ET AL. [18]

Committer's Input: Bit $b \in \{0, 1\}$.

Commitment Phase

1) $\mathsf{C}_{\mathsf{uncon}} \Rightarrow \mathsf{R}_{\mathsf{uncon}}$: Committer sends $(\mathsf{init}_{\mathsf{PUF}}, \mathtt{normal}, \mathsf{sid}, \mathsf{C}_{\mathsf{uncon}})$ to $\mathcal{F}_{\mathsf{PUF}}$ and obtains response $(\mathsf{initialized}_{\mathsf{PUF}}, \mathsf{sid})$. Committer uniformly selects a query $q \in \{0, 1\}^n$ and sends $(\mathsf{eval}_{\mathsf{PUF}}, \mathsf{sid}, \mathsf{C}_{\mathsf{uncon}}, q)$ and receives response $(\mathsf{response}_{\mathsf{PUF}}, \mathsf{sid}, q, a)$. Committer obtains $(st, p) \leftarrow \mathsf{FuzGen}(a)$, and sends p to $\mathsf{R}_{\mathsf{uncon}}$. Committer sends $(\mathsf{handover}_{\mathsf{PUF}}, \mathsf{sid}, \mathsf{C}_{\mathsf{uncon}}, \mathsf{R}_{\mathsf{uncon}})$ to $\mathcal{F}_{\mathsf{PUF}}$.
2) $\mathsf{C}_{\mathsf{uncon}} \Leftarrow \mathsf{R}_{\mathsf{uncon}}$: Receiver receives p' from the committer and $(\mathsf{handover}_{\mathsf{PUF}}, \mathsf{sid}, \mathsf{C}_{\mathsf{uncon}})$ from $\mathcal{F}_{\mathsf{PUF}}$. It uniformly chooses $r \in \{0, 1\}^l$ and sends it to the committer.
3) $\mathsf{C}_{\mathsf{uncon}} \Rightarrow \mathsf{R}_{\mathsf{uncon}}$: If $b = 0$, committer sends $y = st$ to the receiver. Else it sends $y = r \oplus st$.

Decommitment Phase

1) $\mathsf{C}_{\mathsf{uncon}} \Rightarrow \mathsf{R}_{\mathsf{uncon}}$: Committer sends (b, q) to receiver.
2) $\mathsf{C}_{\mathsf{uncon}} \Leftarrow \mathsf{R}_{\mathsf{uncon}}$: Receiver receives (b', q') from the committer and sends $(\mathsf{eval}_{\mathsf{PUF}}, \mathsf{sid}, \mathsf{R}_{\mathsf{uncon}}, q')$ to $\mathcal{F}_{\mathsf{PUF}}$ and obtains $(\mathsf{response}_{\mathsf{PUF}}, \mathsf{sid}, q', a')$. It then computes $st' \leftarrow \mathsf{FuzRep}(a', p')$. If $b = 0$, it checks if $st' = y$. Else, it checks if $st' = y \oplus r$. If the check passes, it accepts, else it rejects.

Part III

Formalization of Physical Unclonable Functions and Unique Objects

Chapter 7

On the Foundations of Physical Unclonable Functions

Having analyzed the usability (and non-usability) of PUFs in advanced cryptographic protocols in Part II, we now turn to foundational aspects of PUFs and their formalization in this third part. Among other things, we deal with the differentiation of various PUF-like primitives, their mathematical, formal definition, and the completion of formal security proofs for PUF protocols.

We start our considerations in this Chapter 7 with the paper:

- U. Rührmair, J. Sölter, F. Sehnke: *On the Foundations of Physical Unclonable Functions.* Cryptology e-Print Archive, Report 2009/277, 2009.

Said work of the candidate is one of the first publications on the formal foundations of PUFs, arguably the first paper solely dedicated to this topic. [1] It extends work that was firstly presented by the candidate in an invited talk at Dagstuhl in 2008 [45].

Among other things, we observe certain conceptual problems with existing PUF definitions in our paper. One of the main issues is the use of asymptotic notions (such as polynomial time) in connection with a single, finite PUF. Secondly, we introduce a game-theoretic approach in formal PUF definitions, which specifies the task of an adversary as a game, and thereby implicitly stipulates an adversarial model. This new approach has been continued in other PUF-definitions, for example by Armknecht et al. in 2011 at the IEEE Symposium on Security and Privacy [2], or by Rührmair et al. in 2010 [55]. We also analyze the entropy and information content in PUFs from a fundamental physical perspective, showing that PUFs cannot contain an amount of entropy that is exponential in their size. Finally, one of our main contributions in this

[1]We emphasize, however, that some early PUF works did contain discussions of certain formal PUF aspects, making some first contributions to the foundations of PUFs: For example, R. Pappu proposes PUF-definitions in his PhD thesis in 2001 [41], and so does B. Gassend in his MSc thesis in 2003 [24]. Also Guajardo et al. lead a semi-formal discussion which touches upon the foundations of PUFs in 2007 [27]. However, it seems fair to say that none of these early works focused solely or in the same level of detail on PUF-foundations as we do in this chapter. Furthermore, we analyze said earlier definitions closely in this chapter, working out a number of problematic aspects of theirs.

paper is to argue that certain subclasses of PUFs should be distinguished in formal treatments. It seems fair to say that this distinction between Weak PUFs and Strong PUFs has made its way into the PUF community in the meantime, and is used in a large number of PUF papers at this date.

According to Google scholar, the work has been cited 102 times to this date [26]. This makes it the most quoted paper of this thesis.

On the Foundations of Physical Unclonable Functions

Ulrich Rührmair Jan Sölter Frank Sehnke

June 10, 2009

Abstract

We investigate the foundations of Physical Unclonable Functions from several perspectives. Firstly, we discuss formal and conceptual issues in the various current definitions of PUFs. As we argue, they have the effect that many PUF candidates formally meet no existing definition. Next, we present alternative definitions and a new formalism. It avoids asymptotic concepts like polynomial time, but is based on concrete time bounds and on the concept of a security experiment. The formalism splits the notion of a PUF into two new notions, Strong t-PUFs and Obfuscating t-PUFs.

Then, we provide a comparative analysis between the existing definitions and our new notions, by classifying existing PUF implementations with respect to them. In this process, we use several new and unpublished machine learning results. The outcome of this comparative classification is that our definitions seem to match the current PUF landscape well, perhaps better than previous definitions. Finally, we analyze the security and practicality features of Strong and Obfuscating t-PUFs in concrete applications, obtaining further justification for the split into two notions.

1 Introduction

Physical One-Way Functions (POWFs), Physical Random Functions (PRFs), and Physical Unclonable Functions (PUFs) are emerging as a new type of cryptographic primitive [1, 2, 3, 5, 6, 7, 10, 4]. While the two former terms have been coined first, today mainly the expression Physical Uncloneable Function or PUF is in use. In a nutshell, a PUF is a mathematical function that is derived from the behaviour of a complex physical object or device. The object can be excited with many possible stimuli C_i, upon which it reacts with corresponding responses R_i. This allows to regard the object's behavior as a function that maps stimuli C_i to responses R_i. Typical applications of PUFs lie in security tasks such as tamper protection, intellectual property protection, read-proof memory, and key obfuscation. They can also be used for classical cryptographic protocols such as key distribution [1] or bit commitment [2].

Despite their broad application spectrum, there are yet no definitions that capture the notion of a PUF consistently and in a contradiction-free manner. Most existing formalizations exhibit problems on the formal and/or on the conceptual side, as we will argue later. Furthermore, it seems that currently more than one concept or notion goes under the term of a "PUF": On the one hand, structures with a highly complex input-output behavior and very many challenges, such as the optical PUF of [1]. On the other hand, structures that have essentially one challenge [6, 10], and which are mainly used for key obfuscation applications or as so-called "POKs" [3].

We take this situation as a reason to investigate the foundations of PUFs further. We start by a thorough formal analysis of currently existing PUF-definitions (section 2). Some problematic aspects are revealed, which are in part caused by the application of concepts like polynomial time and negligible information in the context of PUFs. Subsequently, we propose a new formalism and new definitions (sections 3 and 4). We suggest that the current concept of a PUF should be split into two distinct notions: Strong PUFs and Obfuscating PUFs. Our definitions avoid asymptotic notions, and are based on concrete time bounds and the concept of a security experiment. They build upon several previously existing concepts and definitions [2, 3, 10].

We then go on to classify current PUF implementations with respect to our two new definitions, and also with respect to the previously existing definitions (section 5). In this process, we use several new and unpublished machine learning results that were recently obtained in our group. They show that not only the arbiter PUF, but also improved variants (XOR Arbiter [5, 7], Lightweight PUF [12] and Feed-Forward Arbiter [5]) can all be broken by ML techniques. The resulting classification suggests that our definitions can map the current PUF landscape well, perhaps even more consistently than previous approaches.

Finally, we argue that also from an application perspective, it makes sense to maintain the suggested split into two new notions (section 6). To that end, we analyze the differing security and practicality features that occur in the application of strong and obfuscating PUFs for entity identification. In section 7, we argue that a general emulation of strong PUFs is possible on the basis of obfuscating PUFs, but at the cost of reduced security and practicality. Once more, this strengthens the case for our distinction between strong and obfuscating PUFs. We conclude the publication by a summary in section 8.

2 Problems with Existing Definitions of PUFs

Let us analyze the formal definitions/specifications of PUFs that have been published up to date. For the convenience of the reader, these definitions are provided in the Appendices A, B and C. *Please note* that each of these definitions possesses its own individual merits, and that they are significant pieces of scientific work. Our discussion merely would like to point at the non-trivial obstacles that naturally and necessarily occur in the formulation of a consistent and workable definition of PUFs.

2.1 Physical One-Way Functions

We start with the definition of physical one-way functions [2], which is given in Appendix A. It certainly owns the merit of being the first formalization in the field. But there are some problematic aspects, both on the formal/logical and on the conceptual side, which we list below.

Asymptotic concepts vs. finite function. The definition employs the concept of polynomial resources and of negligible probability, which are familiar from the formalization of mathematical cryptography, for example from the definition of (mathematical) one-way functions [17]. However, these concepts can only be applied in an asymptotic framework, that is, when they are considered for functions with an infinite domain, or for an infinite family of finite functions. If the concepts are applied in finite contexts, logical contradiction arise.

In particular, Definition A.2 considers only one single function f with a finite domain. This has the consequence that *no* function at all can formally fulfill Definition A.2, as the following logical argument shows: Let f be an arbitrary mathematical functions, and suppose, for the sake of contradiction, that f meets Definition A.2. As the definition considers only finite functions, f must be finite, with a finite domain and range. This implies that an algorithm A_f with the following properties can be constructed: (i) A_f incorporates a full look-up table for f. (ii) A_f inverts f by exhaustive search through the domain of the look-up table. (iii) A_f works within a constant time bound (which is essentially the time it takes to browse the look-up table). However, the existence of A_f violates item 2 of the Def. A.2, whence f does not meet the definition, contradiction. This means that no Physical One-Way Function in the sense of Definition A.2 exists.

Functions with small ranges are excluded. Def. A.2 excludes functions with small ranges, for example with the binary output range $\{0, 1\}$. Such functions can be inverted in the sense of item 2 of the definition, as *some* pre-image for one of the two possible function values can be found easily. Many prominent examples of electrical PUFs (arbiter PUF [7], ring oscillator PUF [7], etc.) belong to this class of PUFs, and are excluded by the definition.

Non-invertability is not the relevant security property. Another conceptual argument is that the non-invertability of f is not required in order to make the standard applications of physical one-way functions secure. As an example, consider the two main applications proposed in [1], the forgery-proof labeling of bankcards and key distribution. Speaking in the language of [2] and Notation A.1, they merely require that an adversary without access to the physical system Σ cannot give the correct function value $f(X, P_z)$ for a randomly chosen z, even if he has had previous access to the system Σ for some time.

2.2 Physical Random Functions

Let us now elaborate on the definition of Physical Random Functions (see Appendix B) for comparison. It is certainly very compact and intuitively appealing, and also includes some sort of asymptotic treatment: The notions polynomial and negligible are taken to be relative to the size of the system (see Appendix B). But the definitin nevertheless touches upon some problematic issues.

PUFs with polynomially many challenges are excluded. Definition B.1 allows an adversary to measure polynomially many challenge response pairs (CRPs). This has the consequence that several of the most prominent examples of PUFs will not meet the definition: Since they only possess polynomially many challenges at all, an adversary can create a full look-up table without breaking the polynomial CRP bound, and subsequently use the table for correct PUF prediction. This applies to the ring oscillator PUF proposed in [7], which has only a quadratic number of challenges. It surprisingly holds for the optical PUF of [1], too: Its number of CRPs is directly proportional to the x and y dimensions of the scattering token, multiplied by a constant factor for the finite number of distinct laser angles realizable by the limited measurement set up (this latter number does not grow with the token size). Excluding these PUFs for formal reasons, while especially the optical PUF seems secure by all practical standards, seems problematic.

Information content of physical systems is polynomially bounded. It is worth noting in this context that the amount of information (measured in bits) that can maximally be stored in any physical system S is always polynomially bounded in the size of the system. This can be seen in two ways. Firstly by some simple heuristic: The amount of atoms (or electrons or quarks etc.) in a physical system is polynomially related to the volume. If each of these particles can store a constant number of bits, the information content is polynomial. Even if we speculate that each particle could store a constant (or even a polynomial!) number of bits in its precise relation to each of the other, polynomially many particles (for example by coupling), the overall number of stored bits would still remain polynomial.

There is, however, also a more fundamental argument to the same end. It has been known for a few years that the generalized second law of thermodynamics can be used to derive bounds on the maximal entropy (or information content) of a finite, isolated system. The two most famous bounds are the Bekenstein bound and the holographic bound by t'Hooft and Susskind [20]. The latter states that for any isolated physical system S which fits into a sphere of radius R, the maximal entropy or information content H_S of S is bounded by

$$H_S \leq \pi c^3 R^2 / \hbar G. \tag{1}$$

Thereby c denotes the speed of light, G is Newton's constant, and $\hbar$ is the Planck constant. The equation establishes the polynomial upper bound that we sought.

If the maximal information content of any physical system is upper bounded polynomially in its size (volume/area), however, this makes it questionable whether the usual polynomial/super-polynomial distinction is the right measure to characterize PUFs.

2.3 Physical Unclonable Functions

Another characterization of PUFs was given in [10] (see Description C.1 in Appendix C). It is not formally stated as a definition in the original text, whence we term it as a *description.* It has the great benefit of introducing aspects like measurement noise and also tamper sensitivity. But there are a few problematic aspects related to its specifications.

Asymptotic concepts vs. finite function. It seems reasonable to conclude that a PUF implemented by a finite device is indeed a finite function. This means that in Description C.1, the asymptotic concepts of negligible probability and the notion of a finite function are again mixed. By a similar argument as carried out full detail in section 2.1, this has the consequence that formally no physical unclonable function in the sense of Description C.1 can exist.

Functions with small ranges are excluded. Item 2 of the description states that it must be impossible to come up with the response to a random challenge, except with negligible probability. However, most known electrical PUFs only have one single bit as oputput (e.g. ring oscillator and arbiter PUFs, and also any variants of the arbiter PUF such as feed-forward arbiter, XOR arbiter, or leigthweight PUFs). Hence, an adversary can always guess their output with probability 1/2. This means that all of these PUFs are excluded by the description, which is not desired.

Strong PUFs in the sense of Description C.1 cannot exist. As argued in detail in section 2.2, the information content of an isolated physical system is bounded polynomially in its size (volume/surface area). This means that a strong PUF with an exponential number of challenges in the sense of Description C.1 cannot exist. This is shown by the following argument: Suppose, for the sake of contradiction, that a strong PUF according to Description C.1 exists. As stipulated by the description, such a PUF must possess an exponential number of challenges. At the same time, item 1 of the description states that the responses only give a pairwise negligible information about each other. This implies that the CRPs extract an exponential amount of information from a system that maximally can contain a polynomial amount of information, contradiction. Hence, no strong PUF in the sense of Description C.1 can exist.

Weak PUFs in the sense of Description C.1 are a very restrictive notion. Weak PUFs in the sense of Description C.1 may well exist, but the concept of a weak PUF may turn out to be quite a restrictive notion. As our previous discussion indicates, the only known candidates for a weak PUF are coating PUF and SRAM-based PUFs [6, 10]. Similar to our above discussion, Arbiter PUFs and Ring Oscillator PUFs do not fulfill the requirement that their challenges merely reveal a negligible amount of information about each other, and cannot be weak PUFs.

2.4 Summary

The current definitions of PUFs exhibit problems, which mainly arise from the use of notions like polynomial time or negligible probability in the context of PUFs. Our discussion suggests that these measures should be avoided. They also have problems in formally expressing the unclonability of the system underlying the PUF. In consequence, many current PUF candidates meet none of the existing definitions in a formal manner (see also section 5.1).

3 Strong t-PUFs

This section covers the first of the two new notions suggested in this paper, Strong t-PUFs. We start by some notation.

Notation 3.1 (Measurements). *We use the following notation in order to describe the measurement of an apparatus M on a physical system S: C_i denotes the stimulus or challenge which the measuring apparatus applies to the system. M_{C_i} or $M_{C_i}^S$ denotes the measurement result or the response that is obtained by the apparatus in dependency on C. $\mathbf{C}_M$ denotes the finite set of all possible challenges C_i which M can apply to S.*

Definition 3.2 (Strong t-PUFs)**.** *Let S be a physical system and M be a measuring apparatus, which may be integrated into the system. S is called a* STRONG t-PUF *or simply a t-PUF with respect to M if the following holds:*

1. *It is practically infeasible for the original manufacturer of S to produce another system S' with*

$$M_C^S = M_C^{S'} \quad \textit{for all } C \in \mathbf{C}_M.$$

2. *It is practically infeasible for any cryptographic adversary Eve, who may execute any physical action allowed by the current state of technology and any practically feasible Turing computation, to succeed in the following experiment with a probability greater than 90%:*

   (a) *Eve is given the system S and the measurement apparatus M for a time period t.*

   (b) *Eve is also granted access for time t to a "time-faithful" oracle O, which outputs values M_C^S on input C. Time-faithful means that O produces its output in the same time span that would be required for measuring the response M_C^S on the real system S via use of M.*

   (c) *At the end of the period of length t, Eve must output a physical system S', and access to O is withdrawn from her.*

   (d) *Subsequently, a measurement parameter C_0 is chosen uniformly at random from the set $\mathbf{C}_M$, and is given to Eve. After that, she must answer with a numerical value V_{Eve}.*

   The experiment is called successful if the following holds:

(i) $V_{Eve} = M^S_{C_0}$.

(ii) For all $C \in \mathbf{C}_M$ *it holds that*

$$M^S_C = M^{S'}_C .$$

Thereby the probability is taken over the uniformly random choice of the challenge C_0*, and the random choices or procedures that Eve might employ during steps 2a to 2d.*

Eve's task in the security experiment is depicted schematically in Figure 1. In order to achieve ease of use of our terminology, we will specify a default value for the parameter t.

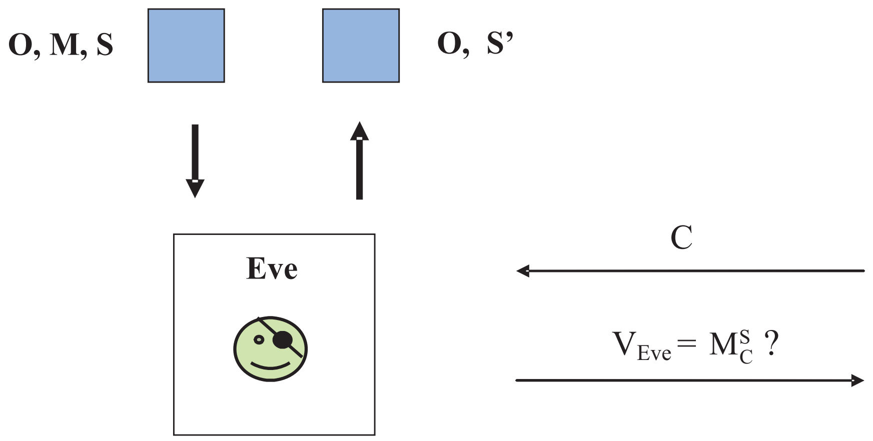


Figure 1: Eve's task in the security experiment of Definition 3.2. She is treated as a black box, and only her input-output behavior is specified.

Notation 3.3 (PUFs). *If* S *is a strong* t*-PUF with respect to some measuring apparatus* M *and with respect to a time period* $t \geq 1$ *day, then we will often call* S *simply a* STRONG PUF *or just a* PUF.

3.1 Discussion

Let us discuss the central features of the definition.

Why S and S'? The reader may ask why we stipulated that Eve must output a system S' after a time period t. It might seem more suggestive to replace item 1. of Definition 3.2 by the following item 1.$'$:

1.$'$ *Eve is given the system* S *and the measurement apparatus* M *for a time period* t*. At the end of that period, Eve must hand back the system* S*.*

Such a statement assumes that Eve leaves the system unchanged, however, thus making a very strong assumption about her internal behavior. Otherwise, i.e. if we formally denote a strongly altered system with different response values still as S, we run into the logical contradiction that $M^S_C \neq M^S_C$ (for at least some $C \in \mathbf{C}$). The cleanest solution to our taste is to differentiate between the input and output object, and to demand that $M^S_C = M^{S'}_C$ for all $p \in \mathcal{P}_M$. Please note that this also condition also must be met by Eve in practical attacks, if she wants her presence to remain undetected.

Why a success probability of 90%? This allows the use of systems with a binary output (that is, 0 or 1) as strong PUFs, since the chance of guessing a single bit value correctly is always at least 1/2. Also, it gives credit to the fact that the natural manufacturing variations in certain systems are not big enough as to change 50% of all responses [4]. Choosing the value 90% instead of 75%, say, is to some extent arbitrary.

Purpose of the oracle O. The main reason for including the oracle O in the definition is to distinguish strong PUFs from systems whose security stems from an access restriction of any kind. Such access restrictions are, for example, present in controlled PUFs, or also in coating PUFs or related systems, where the responses cannot be measured from the outside without tampering the structure. In contrast to that, the security of strong PUFs should only lie in the high physical complexity or disorder of the structure underlying the PUF. PUFs which are secure in the face of such oracle attacks are also automatically secure against any side channel attacks.

All statements relative to M. Our definition stresses the central role of a measuring apparatus in connection with strong PUFs. Being a strong PUF is not a property of the system alone, but of the system *and* the method or apparatus by which the system is measured. This also allows us to easily include the unclonability of a strong PUF in the definition.

Secret key based systems are excluded. Please note that by the manufacturer resistance condition in item 1 of the definition, tamper sensitive systems which contain a secret binary key and use this classical key to generate a complex input-output behavior, are excluded from the definition.

Implications of the Definition. The reader can verify easily that several natural properties of PUFs follow from the definition. For example, the set $\mathbf{C}_M$ must have large cardinality, since otherwise Eve could perform a complete read-out and create a full look-up table even for small values of t. Furthermore, Eve must be unable to machine learn the system, or to clone it physically, in the sense that she can output a second system S^* such that $M_C^S = M_C^{S^*}$ for at least 90% of all $C \in \mathbf{C}_M$.

4 Obfuscating t-PUFs

PUFs have also been suggested for a second type of application, which has been termed "key obfuscation" [3, 6]. The idea is to use the disordered, unique internal structure of a PUF as a non-volatile storage of a secret binary key, whose content can hardly be determined through external, invasive attacks due to its irregular and disordered nature. Typical representatives of this class are PUFs based on SRAM-cells [10] and coating PUFs [6]. Please note that these two PUFs are neither strong PUFs in the sense of Definition 3.2, nor Physical One-Way Functions, Physical Random Functions, or strong Physical Unclonable Functions in the sense of the respective Definitions A.2, B.1, C.1. Therefore we introduce *obfuscating PUFs*. The previous notions that come closest to obfuscating PUFs are weak Physical Unclonable Functions (Definition C.1) and Physically Obfuscated Keys [3].

Definition 4.1 (Obfuscating PUFs)**.** *Let S be a physical system, and M be a measuring apparatus with only one measurement parameter, that is, $\mathbf{C}_M = \{C^*\}$. S is called an* OBFUSCATING t-PUF *for a binary key K_S relative to M if the following holds:*

1. $M_{C^*}^S = K_S$.

2. *The value of K_S results, at least in part, from random, uncontrollable manufacturing variations.*

3. *It is infeasible for Eve to succeed in the following experiment with a probability greater than $(0.9)^{|K_S|}$:*

   (a) *Eve is given S and M for a time period t. She is allowed any action on the systems S and M permitted by current technology.*

   (b) *At the end of that period, Eve must output a binary value V_{Eve}.*

   (c) *The experiment is called successful if $V_{Eve} = K_S$.*

   Thereby the probability is taken over the random choices or procedures that Eve employed during steps 3a and 3b.

Eve's task is illustrated in Figure 2. In order to achieve an easy notation, we introduce the following notational convention.

Notation 4.2 (Obfuscating PUFs, POKs). *If S is an obfuscating t-PUF with respect to some measuring apparatus M and with respect to a time period $t \geq 1$ day, then we will often call S simply an* OBFUSCATING PUF *or a* POK.

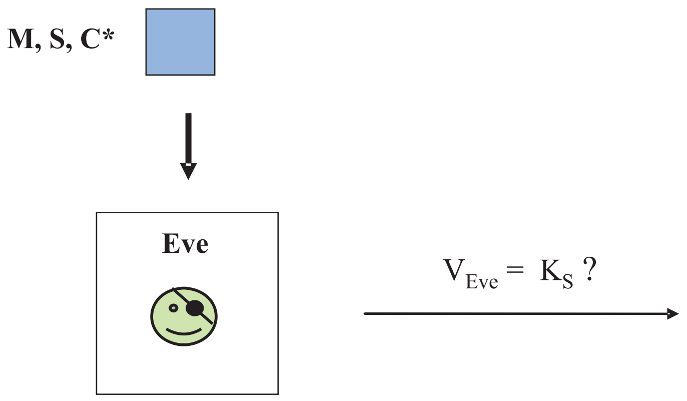


Figure 2: The task that Eve needs to accomplish in order to break the security of an obfuscating PUF

4.1 Discussion

Manufacturer Resistance. We would like to emphasize that we stipulated different types of 'manufacturer resistance' in Definitions 3.2 and 4.1. The requirement expressed in item 1 of Definition 3.2 does not make sense for obfuscating PUFs: Their information content is not necessarily very high. This may allow the manufacturer to fully characterize the system, and/or to extract the obfuscated key K_S during a certain substep of the manufacturing process. Subsequently, he may construct another piece of hardware S', which simply stores K_S in classical, non-volatile form. This system will behave indistinguishably from the original S, violating the original manufacturer resistance stated in item 1 of 3.2. Therefore another type of manufacturer resistance had to be expressed in Def. 4.1.

Only one challenge. Definition 4.1 stipulates that an obfuscating PUF must only have one measurement parameter. The reason is that in practical applications, the hardware system containing the obfuscating PUF always uses it to derive the same key (or the same small set of keys), and proceeds these keys by further cryptoschemes. This implies that some form of fixed challenge must be hardwired into the system. The definition pays respect to this fact.

Distinction to systems with binary keys. The type of manufacturer resistance expressed in item 2 of the definition also distinguishes obfuscating PUFs from classical, tamper sensitive systems that store a protected, secret key. Such systems do not, and should not, meet the definition.

5 Classification

We will now try to classify the main Physical Unclonable Functions that have been proposed to date with respect to our two new notions. Please note that – similar to classical cryptography – we cannot formally prove that a certain candidate meets one of the definitions, we can only *disprove* it. For brevity, we only sketch the respective arguments.

Optical PUF. Due to its high internal complexity and in lack of a known method to reverse engineer or machine learn it, the optical PUF of [1] is a good candidate for a strong PUF (in the sense of Def. 3.2. Since the measurement apparatus and also the measurement process can be observed by Eve, she can derive the measurement angle(s) and point(s) of incidence that are used for the derivation of a potentially obfuscated key K_S. The optical PUF hence does not seem suitable as obfuscating PUF, as long as it is not integrated.

Optical Integrated PUF. An integrated optical PUF has been proposed in [8] Its internal complexity is substantial, but does not seem to be as outstanding as in the case of the original optical PUF [8]. In lack of a proven method to attack it, it must be regarded as a candidate for a strong PUF. Due to its integrated nature, it is also a candidate for an obfuscating PUF.

Ring Oscillator PUF. A ring oscillator PUF (RO PUF) [7] with n ring oscillators can be fully characterized with $n \cdot \log n$ bits of information (i.e. via a list that sorts the n oscillators in ascending order with respect to their frequencies). If the challenges are chosen carefully, a set of challenge-response-pairs of comparable cardinality suffices in order to derive this list. In any case, the maximal set of all possible CRPs has only cardinality $O(n^2)$, making even a full read-out practically feasible. Thus, the RO PUF is no strong PUF. At the same time, it is a candidate for an obfuscating PUF.

Arbiter PUF. Arbiter PUFs [7] have an exponential number (in the number of stages) of possible challenges, meaning that full read-out of all CRPs is impossible. Nevertheless, machine learning of their behavior has been carried out successfully by standard ML methods (support vector machines [5] and even perceptrons [4]). This means that they are no strong PUFs, but they could be used as obfuscating PUFs.

XOR Arbiter PUF. XOR Arbiter PUFs [5, 7] consist of a number of k independent Arbiter PUFs, each with the same number of stages. The same challenge C_i is applied to all of them, and their responses $R_i^1, \ldots, R_i^k$ are XORed with each other to obtain the overall response R_i. [1] They have been introduced to make the known ML attacks on Arbiter PUFs more difficult. One obvious disadvantage is that their read-out stability decreases exponentially in k. Recently, our group has carried out ML experiments on XOR Arbiter PUFs with 4 XORed arbiters. The methods we applied was logistic regression with Rprop gradient descent. The data samples (training set, test set) were generated by an additive linear delay model. As described in greater detail in Appendix C, we achieved prediction rates of 99 %, meaning that the XOR Arbiter PUF can be fully broken by this type of ML attack. Hence, it is no strong PUF, but a candidate for an obfuscating PUF.

Lightweight PUFs. Lightweight PUFs (LW PUFs) have been introduced in [12]. They are, in fact, a special sort of XOR arbiter PUFs, with the only difference that the wiring of the external challenge bits to the internal challenges that are applied to the k arbiters is non-trivial. Several wiring architectures are conceivable, and one particular example is described in [12]. Nevertheless, we could show (see Appendix D) that LW PUFs can be broken by essentially the same methods as XOR arbiter PUFs, namely logistic regression with Rprop gradient descent. Our experiments also proved that not only the original wiring of [12] can be learned, but also random wirings or mappings between the external input and the internal challenges to the k arbiters. This suggests that LW PUFs in their current form are not secure as strong PUFs. They remain candidates for obfuscating PUFs, though.

Feed-Forward Arbiter PUF. Feed-Forward Arbiter PUFs (FF-Arb PUFs) have been introduced, too, to make ML attacks more difficult [5, 7]. Again, one of their disadvantages is their decreased stability; for example, the stability of a FF-Arb PUF with seven FF-loops decreases to 90.16 % if considered over a temperature variation of 45° C [5]. Despite the fact that perceptrons and SVM fail to learn these structures, more involved machine learning techniques are capable of attacking them. Some very recent ML-experiments conducted in our group suggest that FF-Arb PUFs are vulnerable to so-called evolutionary algorithms. These are particularly suited for our task, since they naturally allow us to “feed” or “include” our knowledge about the internal layout or wiring of the PUF in the fitness evaluation step of the ML algorithm. The CRP-data samples were again generated via a linear additive delay model. This means that even FF-Arb PUFs are no strong PUFs, but that they can mainly serve as obfuscating PUFs.

Coating PUF. Coating PUFs [6] are no strong PUFs according to Definition 3.2: Eve can query the oracle that she is provided with for all possible challenge-response-pairs of the coating PUF within very short time (there is essentially just one CRP, which gives the local capacitances determined by all sensors). Please note that without the oracle, Eve would find it hard to determine these capacitances: Any invasive attack would change the capacitance and destroy the responses. In order words, coating PUFs nicely illustrate why it is important to include the oracle access of Eve in Definition 3.2. Coating PUFs are no strong PUFs, but constitute the archetypical candidate of an obfuscating PUF.

[1] Please note that the other obvious option to set up an XOR-based arbiter PUF, namely to apply independent challenges $C_i^1, \ldots, C_i^k$ at the k arbiters, does not make sense: In this case, we can fix all challenges apart from those challenges applied to one of the k arbiters. This creates a structure that behaves essentially like a single arbiter. Then, we can machine learn the behavior of this single arbiter by the known methods [5]. Subsequently, we can go on to fix other challenges and vary the single challenges applied to another arbiter, etc., until we have successively machine learned the behavior of the whole structure.

SRAM-based PUF/Butterfly PUF. By the very same argument as above, SRAM-based PUFs/Butterfly PUFs [10, 11] are no strong PUFs, but candidates for obfuscating PUFs.

Summary. The following table summarizes our findings. As noted earlier, we cannot positively prove that one of the listed PUFs actually is an obfuscating or strong PUF according to Definitions 3.2 and 4.1. They can merely have the status of candidates. On the other hand, one can disprove that a PUF fulfills Definitions 3.2 or 4.1, leading to *"no"*-entries in the table. The table shows that with one exception, all currently existing "PUFs" are either candidates for strong PUFs or obfuscating PUFs, but not for both, and that also every known PUF is a candidate for at least one of the two notions.

	Opt.	**Int.Opt.**	**R.Osc.**	**Arb.**	**XOR**	**LW**	**FF**	**Coat.**	**SRAM**
Obf. PUF	No	Cand.	Cand.	Cand.	Cand.	Cand.	Cand.	Cand.	Cand.
Str. PUF	Cand.	Cand.	No	No	No	No	No	No	No

5.1 Comparison with Previous Notions

For comparison, we also provide a classification of current PUF candidates with respect to the previous notions of Physical One-Way Functions (POWF, Def. A.2), Physical Random Function (PRF, Def. B.1), Strong Physical Unclonable Functions (Strong PUF as in Def. C.1) and Weak Physical Unclonable Functions (Weak PUF, Def. C.1). As the reader may verify relatively easily, the entries in the following table follow from the discussions in section 2 and 5, and from the new machine learning results presented in Appendix D.

Please note that it is not fully clear how Def. B.1 should be interpreted with respect to Coating PUFs and SRAM-based PUFs, whence we leave the respective classification open.

	Opt.	**Int.Opt.**	**R.Osc.**	**Arb.**	**XOR**	**LW**	**FF**	**Coat.**	**SRAM**
POWF	No	No	No	No	No	No	No	No	No
PRF	No	Cand.	No	No	No	No	No	??	??
Strong PUF	No	No	No	No	No	No	No	No	No
Weak PUF	No	No	No	No	No	No	No	Cand.	Cand.

The two last tables seem to support our proposed formalism of strong t-PUFs and obfuscating t-PUFs, and the split of PUFs into two different notions.

6 Differing Security and Practicality Features in Applications

We will now analyze the differing properties of strong PUFs and obfuscating PUFs in a concrete security application. We chose an identification-like scenario that was among the first applications ever suggested for PUFs [1, 2]. Strong PUFs and obfuscating PUFs have notable differences therein.

6.1 Setting

The situation we consider is as follows: Alice wants to identify herself towards a central server via a certain physical token that she holds. The token is put into a terminal, which communicates with the server. This basic setting applies in many practical situations, for example bank cards, branded products, identity cards, access cards, etc. [1, 2]. The application of PUFs to both scenarios has been described intensively in the literature.

6.2 Review of Basic PUF-Protocols for Identification and Labeling

Before we analyse the differences, we will quickly repeat the basic protocols and schemes for the convenience of the reader. Readers familar with the protocols may skip to section 6.3.

If strong PUFs are applied to the above identification setting, the protocol works as follows [1]: The central server must know a list of CRPs. It sends randomly chosen challenges C_i from the list to the terminal, which returns the responses M_{C_i} obtained from the label. These are compared to the CRPs in the server's list. Any CRP cannot be used more than once in this setting, meaning that the server must have a very large list on stock. In order to reduce the storage requirements, protocols that allow refreshment of the CRPs at the central server are possible [8].

In the case of an obfuscating PUF S that contains a key K_S, two possibilities are conceivable. Option one is that the central server stores K_S, and that the central server executes a symmetric identification protocol on the basis of K_S with the labeled item. The communication between them is executed via the terminal.

Option two is that the labeled item stores a public key/private key pair (pk, sk). The private key sk is encrypted with the obfuscated key K_S via a one time pad scheme, and the encrypted value $Enc(sk)$ is stored on the labeled item. The public key pk is stored in plain.

In order to authenticate (pk, sk), the public key pk must be signed (or 'tagged') by some trusted authority CA via its key $SigKey_{TA}$. The terminal holds the corresponding public verification key $VerKey_{TA}$. The verification of a label by the terminal proceeds in two steps: (i) The labeled item passes on the CA's 'tag' and the key pk to the terminal. (ii) The terminal verifies the tag via use of $VerKey_{TA}$. (iii) The terminal executes an asymmetric identification protocol with the labeled item on the basis of pk. During this protocol, the labeled item decrypts $Enc(sk)$, and uses sk to generate its responses. This setting allows even offline-verification of the label, i.e. without a connection to a central server.

6.3 Practicality Features

We will now illustrate the different practicality features of Strong and Obfuscating PUFs in the above example application.

Strong PUFs. Strong PUFs only allow secure labeling in connection with a central server, which must store a very large number of CRPs. The computational load on the labeled item is essentially zero, meaning that very cheap labels can be generated. Error correction on the measurement values can be carried out by the terminal.

Refreshment of CRPs is possible via a protocol given in [8], which reduces the storage requirements on the server. However, the protocol comes at the cost of reduced security properties (see section 6.4).

Obfuscating PUFs. Obfuscating PUFs allow verification of the label via a central server, too, but advantageously on the basis of one single short bitstring stored on the server. This string can be used multiple times and does not need to be refreshed. The label must execute a symmetric identification scheme, imposing some computational load on it. Alternatively, as described in section 6.1, even offline verification of the labels is possible. In this case, however, asymmetric schemes must be executed by the label itself, leading to a strong computational load on a mobile system.

6.4 Security Features

The different security features of Strong and Obfuscating PUFs in the above appliance are as follow.

Strong PUFs. Strong PUFs lead to secure labeling under the mere assumptions that the underlying PUF is secure. The scheme remains secure even if Eve has physical access to the labeled item and the PUF contained therein multiple times.

If the CRPs at the server are refreshed via the protocol of [8], one necessary assumption is again the security of the underlying strong PUF. The protocol brings about a number of additional assumptions, however: Firstly, unproven computational assumptions on the security of the cipher employed in the protocol (see [8]). Secondly, it can be shown that the protocol is not secure against an adversary who (i) eavesdrops the communication between the item, the terminal and the server, and who (ii) gains physical access to the PUF more than once. This enforces the slightly unrealistic assumption that Eve must not have access to the PUF more than once.

Obfuscating PUFs. Obfuscating PUFs allow secure labeling only under a number of additional assumptions, which go beyond the mere security of an obfuscating PUF. Let us start with the case of a central sever which stores K_S: We firstly must assume that the underlying MAC is computationally secure. Next, we face two hardware assumptions *on the labeled item*: We must suppose that the transfer from the key K_S to the unit where the MAC is computed on the item can be carried out securely, without allowing Eve invasive, semi-invasive or side channel attacks. Finally, we must assume that the MAC is not only computationally secure, but that it is also implemented securely in hardware on the item, and that no power analysis, emanation analysis or other side channel attacks are possible. This host of assumptions arises because obfuscating PUFs eventually are nothing else than a secure form of secret key storage. Their practical use has to be complemented by classical mathematical cryptoschemes, which bring about the standard, well-known attacks.

The security analysis in the case of offline labeling, where we do not use a central server, but employ a central tagging authority and a public key/private key pair stored on the item, is similar.

6.5 Summary of Practicality and Security

Strong PUFs can lead to very nice security properties under very few security assumptions. Obfuscating PUFs, to the contrary, require more security assumptions, and can lead to a signficant computational load on mobile systems. On the good side, they sometimes offer practicality advantages, such as offline label verification in our above example. Our discussion reassures the differences of strong and obfuscating PUFs also on the application side.

7 General Emulation: Strong from Obfuscating?

Let us discuss one final aspect in the relation between strong and obfuscating PUFs. It seems at first glance that a strong t-PUF S' can be constructed from an obfuscating t-PUF S in the following way: S' includes S as a subsystem. Whenever S' is presented with a challenge C, it derives the key K_S from S. Then, it computes and outputs $PRNG(C, K_S)$, for some suitably chosen and implemented pseudo-random number generator $PRNG$.

The construction seems to bring about a general simulation theorem, but nevertheless, there are a few problems associated it. First of all, it requires a number of additional assumptions. They have already been touched on in section 6.4:

1. The PRNG must be computationally secure (as a mathematical function).
2. The PRNG must be implemented physically in a secure manner. In particular, no side channel attacks must be possible (power consumption, emanation, or timing analysis, etc).

3. The on-circuit transport of K_S from the obfuscating PUF to the PRNG must be handled securely.

4. The underlying obfuscating PUF must be secure according to Def. 4.1.

Furthermore, from a formal perspective, the construction does not fulfill the manufacturer resistance condition in item 1 of Def. 3.2. The manufacturer could read out K_S, copy it and use it in several systems, which would show indistinguishable behavior.

Finally, the above construction requires that the PUF can handle the computational load of the PRNG. This practicality restriction, together with the above list of additional assumptions and the mentioned definitional issues, makes it preferable in any case to directly construct strong PUFs, even though our construction shows that they can to some extent be derived from obfuscating PUFs.

8 Summary and Conclusions

Our topics in this paper have been as follows:

(i) We discussed some logical and conceptual issues in the existing definitions of PUFs.

(ii) We proposed to split the current notion of a PUF into two new notions, strong PUFs and obfuscating PUFs. Strong PUFs try to express the original concept of a POWF/PRF in a consistent manner [2, 3], while the purpose of obfuscating PUFs is to pinpoint those properties relevant for key obfuscation [3].

(iii) We suggested a new framework for these two notions, and presented corresponding formal definitions.

(iv) We provided classifications of existing PUFs with respect to our new framework and with respect to previous definitions.

(v) In this classification, we used some new machine learning results of our group. For the first time, these results show that XOR-Arbiter and Feed-Forward Arbiter PUFs can be attacked on large scales.

(vi) We illustrated important security and practicality differences between strong PUFs and obfuscating PUFs in practical applications.

(vii) We discussed an emulation strategy by which strong PUFs can (or cannot) be constructed from obfuscating PUFs.

Our findings also point to the fact that even though strong PUFs are a powerful and elegant security tool, currently no integrated electrical implementations exist. The investigation of such structures seems to be a rewarding future research goal.

Acknowledgements

The authors would like to thank Qingqing Chen, György Csaba, Jon Finley, Stefan Katzenbeisser, Heike Busch, Paolo Lugli, Christian Osendorfer, Jürgen Schmidhuber, Ulf Schlichtmann, Vera Stoyanova, Martin Stutzmann, and Ju Xueming for enjoyable and helpful discussions.

References

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

[2] R. Pappu, *Physical One-Way Functions*, PhD Thesis, MIT, 2001.

[3] Blaise Gassend, *Physical Random Functions*, MSc Thesis, MIT, 2003.

[4] Blaise Gassend, Daihyun Lim, Dwaine Clarke, Marten van Dijk, Srinivas Devadas: *Identification and authentication of integrated circuits.* Concurrency and Computation: Practice & Experience, pp. 1077 - 1098, Volume 16, Issue 11, September 2004.

[5] Daihyun Lim: *Extracting Secret Keys from Integrated Circuits.* MSc Thesis, MIT, 2004.

[6] Pim Tuyls, Geert Jan Schrijen, Boris Skoric, Jan van Geloven, Nynke Verhaegh, Rob Wolters *Read-Proof Hardware from Protective Coatings.* CHES 2006: 369-383

[7] G. Edward Suh, Srinivas Devadas: *Physical Unclonable Functions for Device Authentication and Secret Key Generation.* DAC 2007: 9-14

[8] P. Tuyls, B. Skoric. *Strong Authentication with PUFs.* In: Security, Privacy and Trust in Modern Data Management, M. Petkovic, W. Jonker (Eds.), Springer, 2007.

[9] P. Tuyls, B. Skoric, T. Kevenaar (Eds.): *Security with Noisy Data.* Springer 2007.

[10] Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, Pim Tuyls: *FPGA Intrinsic PUFs and Their Use for IP Protection.* CHES 2007: 63-80

[11] Sandeep S. Kumar, Jorge Guajardo, R. Maes, Geert Jan Schrijen, Pim Tuyls: The Butterfly PUF: Protecting IP on every FPGA. HOST 2008: 67-70.

[12] Mehrdad Majzoobi, Farinaz Koushanfar, Miodrag Potkonjak: *Lightweight secure PUFs.* ICCAD 2008: 670-673.

[13] T. Bäck: *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms.* Oxford University Press, USA, 1996.

[14] H.P. Schwefel: *Evolution and optimum seeking.* Wiley, New York, 1995.

[15] I. Rechenberg: *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Fromman-Holzboog, Stuttgart, Germany, 1973.

[16] H.-P. Schwefel: *Numerical Optimization of Computer Models.* John Wiley and Sons, LTD, 1981.

[17] O. Goldreich, The Foundations of Cryptography. Volume 1 & 2, ISBNs: 0-521-79172-3 (Vol. 1)& 0-521-83084-2 (Vol. 2), Cambridge University Press, 2001/2004.

[18] M. Riedmiller, H. Braun: *RPROP-A fast adaptive learning algorithm.* Proc. of ISCIS VII), Universitat, 1992

[19] C.M. Bishop: *Patern recognition and machine learning.* Springer New York, 2006

[20] Jakob D. Bekenstein: *How does the Entropy/Information Bound Work?* Foundations of Physics, vol. 35, issue 11, pp. 1805-1823. Latest version also from http://arxiv.org/abs/quant-ph/0404042.

A Physical One-Way Functions

The notion of a "Physical One-Way Function" has been defined by R. Pappu in [2]. It definitely owns the merit of being the first formalization attempt in the field. Before giving his definition, Pappu introduces some notation, which was developed with hindsight to his optical PUF [1, 2].

Notation A.1 (Notation for Physical One-Way Functions, as in [2]). *Let Σ be a physical system in an unknown state $X \in \{0,1\}^l$. X could also be some property of the physical system. l is a polynomial function of some physical resource such as volume, energy, space, matter, et cetera.*

Let $z \in \{0,1\}^k$ be a specific state of a physical probe P such that k is a polynomial function of some physical resource. Henceforth, a probe P in state z will be denoted by P_z.

Let $y = f(X, P_z) \in \{0,1\}^n$ be the output of the interaction between system Σ containing unknown state X and probe P_z.

On the basis of this notation, [2] devises the following definition.

Definition A.2 (Physical One-Way Functions, as in [2]). *$f : \{0,1\}^l \times \{0,1\}^k \to \{0,1\}^n$ is a* PHYSICAL ONE-WAY FUNCTION *if*

- *$\exists$ a deterministic physical interaction between P and Σ which outputs y in $O(1)$, i.e. constant, time.*
- *Inverting f using either computational or physical means requires $\Omega(exp(l))$ queries to the system Σ.*

  This may be restated in the following way: The probability that any probabilistic polynomial time algorithm or physical procedure A' acting on $y = f(X, P_r)$, where $y \in \{0,1\}^n$ is drawn from a uniform distribution, is able to output X or P_r is negligible. Mathematically,

  $$Pr[A'(f(X, P_r)) \text{ outputs } X \text{ or } P_r] < \frac{1}{p(l)}$$

  where $p(\ )$ is any positive polynomial. The probability is taken over several realizations of r.

We also stipulate that for any physical one-way function f

- *Simulating y, given X and P, requires either $O(poly(l))$ or $O(exp(l))$ in time/space resources depending on whether f is a* WEAK *or* STRONG *physical one-way function.*
- *Materially constructing a distinct physical system Σ' such that its unknown state $X' = X$ is hard.*

B Physical Random Functions

Gassend [3] provides another definition, which does not emphasize the non-invertability of PUFs, but their unpredictability or "randomness".

Definition B.1 (Physical Random Functions, quoted from [4]). *A* PHYSICAL RANDOM FUNCTION *(PUF) is a function that maps challenges to responses, that is embodied by a physical device, and that verifies the following properties:*

1. *Easy to evaluate: The physical device is easily capable of evaluating the function in a short amount of time.*
2. *Hard to predict: From a polynomial number of plausible physical measurements (in particular, determination of chosen challenge-response pairs), an attacker who no longer has the device, and who can only use a polynomial amount of resources (time, matter, etc.) can only extract a negligible amount of information about the response to a randomly chosen challenge.*

In the above definition, the terms short and polynomial are relative to the size of the device.

C Physical Unclonable Functions

Finally, Tuyls et al. give a third specification of PUFs [10]. It is not marked as a definition in their text, whence we quote it as a "description" here.

Description C.1 (Physical Unclonable Functions, quoted from [10])**.** *Physical Unclonable Functions consist of inherently unclonable physical systems. They inherit their unclonability from the fact that they consist of many random components that are present in the manufacturing process and can not be controlled. When a stimulus is applied to the system, it reacts with a response. Such a pair of a stimulus C and a response R is called a challenge-response pair (CRP). In particular, a PUF is considered as a function that maps challenges to responses.*

The following assumptions are made on the PUF:

1. *It is assumed that a response R_i (to a challenge C_i) gives only a negligible amount of information on another response R_j (to a different challenge C_j) with $i \neq j$.*
2. *Without having the corresponding PUF at hand, it is impossible to come up with the response R_i corresponding to a challenge C_i, except with negligible probability.*
3. *Finally, it is assumed that PUFs are tamper evident. This implies that when an attacker tries to investigate the PUF to obtain detailed information of its structure, the PUF is destroyed. In other words, the PUF's challenge – response behavior is changed substantially.*

We distinguish between two different situations. First, we assume that there is a large number of challenge response pairs (C_i, R_i), $i = 1, \ldots, N$, available for the PUF; i.e. a strong PUF has so many CRPs such that an attack (performed during a limited amount of time) based on exhaustively measuring the CRPs only has a negligible probability of success and, in particular, $1/N \approx 2^{-k}$ for large $k \approx 100$. We refer to this case as strong PUFs. If the number of different CRPs N is rather small, we refer to it as a weak PUF. Due to noise, PUFs are observed over a noisy measurement channel i.e. when a PUF is challenged with C_i a response R'_i which is a noisy version of R_i is obtained.

D Machine Learning Attacks on Physical Unclonable Functions

It has long been known that PUFs can potentially be attacked via machine learning (ML) techniques. In these attacks, an adversary collects many CRPs from the PUF and uses them to train an ML algorithm. If successful, the trained algorithm can subsequently imitate the PUF, thereby breaking its security. For example, it has been shown relatively early that perceptrons [4] and Support Vector Machines (SVMs) [5] suffice to break standard arbiter PUFs.

In order to avoid these attacks, strengthened versions of the arbiter PUF have been suggested: XOR arbiter PUFs [5, 7], lightweight PUFs [12] and feed-forward arbiter PUFs [5, 7]. Nevertheless, we were able to successfully break all these variants by suitable ML techniques.

Logistic Regression, XOR Arbiters and Lightweight PUFs. Logistic regression is a well established, generic ML method [19]. As our recent efforts show, it can be applied successfully to the cryptanalysis of the XOR arbiter and the lightweight PUF.

As a starting point, we used the standard linear model of the arbiter PUF described in [5]. In this model, it is assumed that the challenge $\vec{C}$ of a b-stage arbiter PUF is a string from the set $\{-1,1\}^b$ (instead of $\{0,1\}^b$). Correspondingly, the arbiter output is encoded into a binary response $t \in \{-1,1\}$ (instead of $\{0,1\}$). Under these provisions, the final output t can – in dependence of the final propagation delay difference Δ at the end of the sequence of multiplexers – be written as

$$t = sgn(\Delta) = sgn(\vec{w}^T\vec{P}), \tag{2}$$

$$\text{where} \quad P^i = \textstyle\prod_{j=i}^{b} C^j, \quad \vec{P} = (P^1, P^2, \ldots, P^n, 1)^T, \tag{3}$$

and where the parameter vector $\vec{w}$ is a direct function of the (unknown) inner subdelays of the PUF. For the details, see [5].

In order to apply the above model to an XOR-arbiter structure, in which the final output t_{xor} is computed as the parity of k single arbiter outputs, one writes:

$$t_{xor} = sgn(\prod_{r=1}^{k} t_r) = sgn(\prod_{r=1}^{k} \Delta_r) = sgn(\prod_{r=1}^{k} \vec{w}_r^T\vec{P}_r). \tag{4}$$

From the above formalism, we can derive an obvious ML decision boundary by

$$\prod_{r=1}^{k} \vec{w}_r^T\vec{P}_r = 0. \tag{5}$$

Logistic regression now progresses by optimizing the vectors $\vec{w}_r$ in (5) such that the likelihood of observing the CRPs of a training set $\mathcal{M}$

$$p(\mathcal{M}|\vec{w}) = \prod_{m\in\mathcal{M}} \sigma(t^m \cdot \prod_{r=1}^{k} \vec{w}_r^T\vec{P}_r^m) \tag{6}$$

becomes maximal. There are various methods to carry out this optimization. Our experiments showed that Rprop gradient descent [18] performs best, and that standard gradient descent and also iterative reweighted least squares do not work for the considered problem.

This leads to a prediction rate on test sets of 99% for arbiter PUFs, provided that the vectors $\vec{w}_r$ have been well trained on sufficiently large training sets. Empirically, a first estimate for the required number of CRPs is $50 \cdot k \cdot (b+1)$ or $O(kb)$ (as above, k denotes the number of arbiters and b the arbiter length). The vectors $\vec{w}_r$ are considered as well trained if the corresponding decision boundary (5) correctly classifies over 99% of the training set.

However, the algorithm does not yield well trained vectors for every initial choice of the $\vec{w}_r$. Therefore it has to be evaluated repeatedly on the training set, with randomly initialized parameter vectors $\vec{w}_r$, until an adequate solution is found. By this approach the security of the XOR Arbiter PUF and/or the Lightweight PUF with different internal wirings can be broken, as shown below in Table 1. All described PUFs are of type 4-XOR, 64 bit.

	equal challenges	**random challenges**	**lightweight challenges**
training set size [CRPs]	19000	19000	49000
mean computing time [min]	0.7	58.5	12
prediction rate on test set	99.1%	98.9%	98.7%

Table 1: Performance of the logistic regression approach for different XOR and Lightweight PUFs. The examined PUFs are all XOR-arbiter type PUFs, but differ by the precise mapping of the external challenge to the internal challenges applied at the individual internal arbiter PUFs. In the case of equal challenges, the challenge for all the individual internal arbiter PUFs is the same as the external challenge. For random challenges, the overall, external challenge is (pseudo) randomly transformed to individual challenges of the 4 internal arbiters. The mapping of the lightweight PUF is described in [12]. The mean computing time was determined on a standard quad core PC.

Evolution Strategies and Feed-Forward Arbiters. Evolution Strategies (ES) [14] belong to a class of ML techniques called Evolutionary Algorithms. They are inspired by the biological evolution of a population of individuals under certain environmental conditions. The population repeatedly undergoes the evolutionary steps of evaluation, environment selection, partner selection, recombination and mutation, until inividuals are found which match some previously fixed environmental conditions very well.

In our case, an individual is given by a concrete instantiation of the runtime delays in a PUF (or by the vector $\vec{w}$ from equation (5)). The environmental fitness is determined by how well this individual (re-)produces the correct CRPs of the target PUF as output. The outputs of the individual are computed by a linear additive delay model from its subdelays (or from $\vec{w}$), and are compared to several known outputs of the target PUF structure. The best individuals are selected. In the following recombination step, the remaining individuals mutually exchange part of their 'genome'/their individual subdelays, in order to form descendants. In the final mutation step, some of the subdelays are varied randomly, and the process starts anew, producing more and more *'generations'* of the population. Evolutionary algorithms bear the advantage that our full knowledge of the PUF architecture can be exploited by building a corresponding model into the fitness evaluation step.

We used a standard implementation of ES with the ES standard meta-parameters [13]: Population size of (6,36), comma-best-selection, and a global mutation operator of $\tau = \frac{1}{\sqrt{(n)}}$. As the training data underlying the experiments, we used a set of 50.000 CRPs with random subsets of 2.000 CRPs for the evaluation step of the individuals. These CRPs were generated on the basis of a linear additive delay model. The subdelays in the stages were drawn according to a uniform distribution with parameters motivated by the standard fabrication delays that may occur in such a structure. The architecture of the examined structures is shown in Figure 3.

Our ML results are depicted in Figure 4. For each FF-Arb PUF, it shows the best out of 10 runs that were conducted on the structure. The x-axis displays the number of generations, the y axis the prediction error. As shown, in all but one case we have been able to successfully learn the PUFs with rates better than 95 % in only 600 generations. Since ES are a probabilistic method, additional runs can be expected to yet further improve the prediction accuracy. Also other meta-parameters like bigger population sizes, more generations or smaller τ can optimize the performance further. Please note that our accuracy is significantly better than the stability of a FF-arbiter with 7 FF-loops under a temperature variation of 45°C, which is only 90.16 % [5]. In this sense, our experiments indicate that the FF arbiter can be fully broken by ES.

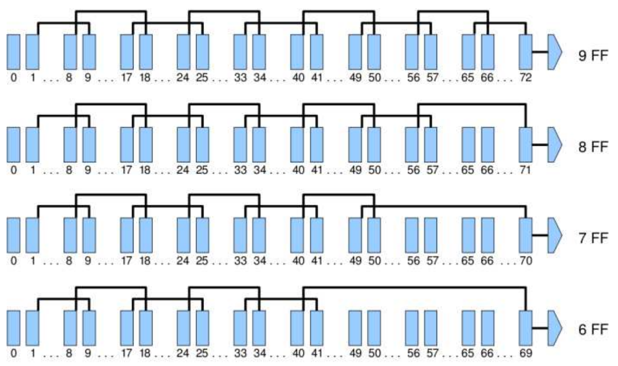


Figure 3: The feed-forward architectures we used in our learning experiments.

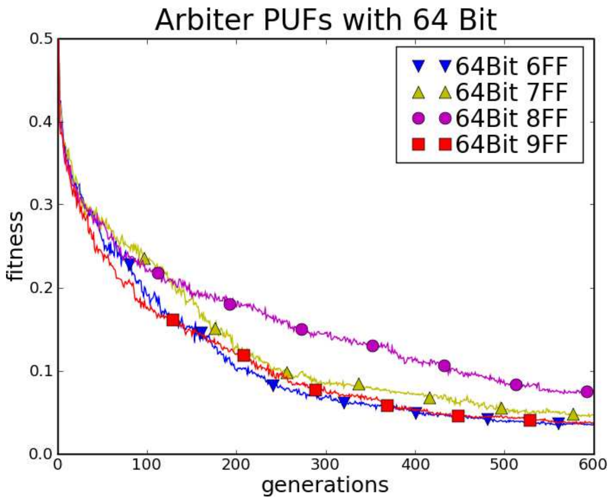


Figure 4: The best of 10 runs on each of the architectures.

Chapter 8

Physical Turing Machines and the Formalization of Physical Cryptography

Some previous work on the formalization of PUFs by Rührmair, Busch and Katzenbeisser [55] assumes that any adversaries who hold physical possession of the PUF are limited to reading out challenge-response pairs; this assumption is called the *"digital attack model" (DAM)* in [55]. The DAM is indeed well motivated by situations where the PUF can only be accessed via its CRP interface, and where this interface cannot be circumvented due to the PUF's natural tamper-sensitivity, as argued in [55].

In practice, however, it is very well conceivable that sophisticated attackers will make arbitrary physical measurements on the PUF, which might be vastly more powerful than mere CRP measurements. This particularly applies to PUFs that do not have a dedicated CRP-interface (such as optical PUFs), or to situations in which well-equipped adversaries can tamper the PUF and circumvent the CRP-interface. It also holds whenever attackers can make side-channel measurements on the PUFs, i.e., whenever they extract helpful information beyond standard CRPs. In fact, the practical viability of such measurements has recently been demonstrated in silicon PUFs [78]. In such scenarios, the DAM from the last chapter is no longer sufficient. From a strictly fundamental perspective, this observation actually points to a gap in the DAM as well as in the PUF-formalisms suggested recently by other authors at major conferences like CRYPTO 2011 and EUROCRYPT 2013 [6, 40].

The above problem is non-trivial to overcome, though. In order to resolve it, a new, formal computational model must be developed, which allows the adversaries and honest users to carry out arbitrary physical actions on the PUF — a fact that renders the classical Turing machine model unsuited. At the same time, the new model must have enough structure to allow formal security proofs, in particular reductionist security proofs. The approach taken by Armknecht, Maes, Sadeghi, Standaert, and Wachsmann at IEEE S&P 2011 [2], where no such model for the adversary is specified, to us therefore appears insufficient to this end.

This chapter now presents a method to resolve the above problems. It introduces a formal computational model that can incorporate arbitrary physical actions, namely so-called *"physical Turing machines (PhTMs)"*, and uses them for a truly "physical" security proof of a certain example scheme. This scheme is not directly based on PUFs, but on unique objects (UNOs) as introduced in Chapter 2.

The choice of UNOs as an example primitive in this chapter has three reasons: Firstly, and perhaps most importantly, the exact security features of UNOs are much stronger related to their physical unclonability than in the case of PUFs, where, for example, also purely numeric and digital modeling attacks pose a threat to security. The same type of purely numeric attack is useless in the case of UNOs; the only viable adversarial strategy for them is to fabricate a true physical clone of some sort. Secondly, our choice allows us to illustrate the effectiveness of our techniques in an extended context beyond classical PUFs, showing that the areas of physical cryptography and disorder-based security are larger than just PUFs. Finally, the presented labeling scheme is a typical example where classical computational assumptions (namely the security of the employed digital signature scheme) and physical assumptions (namely the unclonability of the employed unique object) are combined. PhTMs shine particularly in such a context, since they have the unique feature that they can capture and formalize both types of asymptotic assumptions in one machine model.

We believe that the suggested concepts of a physical Turing machine and of a *"technology"* on which it operates will prove useful beyond PUFs, for example in the formal treatment of other hardware security features. The chapter thus presents a potential outlook on future PUF formalization, making a well-suited concluding chapter of the technical part of this thesis.

The paper that we use in this chapter is:

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

According to Google scholar, it has been cited 6 times to this date [26].

Physical Turing Machines and the Formalization of Physical Cryptography

Ulrich Rührmair
Technische Universität München
80333 München, Germany
ruehrmair@in.tum.de

September 18, 2006

(With Revisions in Introductions and Summary in 2011 and 2014)

Contents

Abstract

In this paper, we discuss how *physical* actions of cryptographic adversaries and *physical* features of security hardware can be formally modeled, and how mathematical proofs involving such *physical* security features can be led. To this end, we introduce two new concepts: Firstly, the notion of a *"physical Turing machine" (PhTM)*, which is a classical Turing machine with the amended ability to process and to act upon physical objects. Secondly, the concept of a *"technology"*, which is a set of admissible physical actions that the PhTM can execute. The power of a PhTM is, in other words, subject to the technology upon which it operates — similar to the power of real-world adversaries, which depends on the technology and machinery that they have available.

We then illustrate the usefulness of our new concepts by two examples: Firstly, we briefly sketch their application in a comprehensive formal treatment of classical cryptography, where they can model adversaries that use inherently *physical* computations to attack security schemes, for example quantum computers [35], optical techniques [34, 17], or similar *physical* methods that might outperform classical Turing machines. The current formalization of classical cryptography, which is merely based on standard Turing machines, does not sufficiently include such adversaries, so we argue. We sketch how this conceptual gap can be closed by use of PhTMs and technologies.

Secondly, we deal in greater detail with the application of PhTMs to so-called physical unclonable functions and variants thereof; this application is actually the main topic of our paper. In this context, we lead a reductionist security proof for a recent method of creating forgery-proof physical "labels" or "tags"' for valuable items. This method combines disordered, unclonable physical structures with classical digital signatures to create unforgeable "labels" that (unlike RFID-tags) do *not* contain any secret keys. We formally show by a reductionist method that the labels are secure as long as the employed digital signature scheme is secure, and as long as the used disordered structures are unclonable. Prior to the proof, we formally express all relevant features and notions, such as the concept of a unique, unclonable physical object, or of a "labeling scheme", via the use of PhTMs, illustrating their power and broad applicability.

We believe that the concepts of PhTMs and technologies will have widespread other applications beyond the two above examples. They could be used for formally expressing side channel resilience and other physical features of classical security systems, or for the development of a form of "physical" structural complexity theory relative to an available physical technology.

1 Introduction and Overview

One recently emerging trend in cryptography and security is to use disordered, randomly structured physical systems in security applications. The approach has often been referred to as *"physical unclonable functions (PUFs)"* [12], sometimes also as *"disorder-based security"* [26] or *"physical cryptography"* [28]. It is driven by the insight that not only the mathematical properties of certain functions (such as non-invertability, pseudo-randomness, etc.) can be exploited cryptographically, but also the analog properties of certain physical objects [22, 12].

One central concept in the area is the phenomenon of *"physical unclonability"*, i.e., the observation that the inevitable, random variations and imperfections in each mesoscopic and macroscopic physical object cannot be cloned or reproduced perfectly. Interestingly, this does not only hold for (well-equipped) adversaries, but even for the original manufacturer of the object: Even he cannot produce two specimen that are exactly the same. This allows new identification mechanisms for secure hardware, or new methods for the unforgeable tagging of valuable items [12, 22, 14].

A second example concept is that certain randomly disordered physical objects can contain a very high information content or entropy. If suitably fabricated, solid-state objects may even contain so much random information that it is impossible to read-out this information completely [29]. Everyone holding the object can only read-out and henceforth know a small, individual fraction of the information. This situation of partial knowledge can then be exploited cryptographically, for example in the context of secure multi-party computation or oblivious transfer [24].

It is well-known that the above disorder-based approach has a number of potential upsides [26]. First of all, it can allow a better protection of secret cryptographic keys in secure hardware. Instead of storing the keys in standard digital memory, they are derived from, or hidden in, the complex analog characteristics of a randomly structured medium. This can make them harder to read out or otherw ise obtain for the adversary [26]. Secondly, the utilization of physical disorder can sometimes avoid standard unproven assumptions in cryptographic protocols, such as the assumed hardness of factoring and computing discrete logarithms. The latter are substituted by alternative hypotheses on the employed physical objects, for example their information content/entropy, their unclonability, or the complexity of their input/output behavior. This creates an alternative fundament for cryptography, one that is independent from the abovementioned classical assumptions. In this sense, disorder-based cryptography could be seen in alignment with other, well-known alternative approaches, for example quantum cryptography [4], post-quantum cryptography [7], noise-based cryptography [20], or the bounded storage model [19]. In this context, disorder-based schemes for oblivious transfer, key exchange, bit commitment, and multi-party computation have been suggested and analyzed [24, 5, 21].

In recent years, the mathematical formalization of PUFs and disorder-based approaches has attracted increasing attention within the community, leading to publications at major conferences such as CRYPTO, EUROCRYPT, IEEE S&P, and elsewhere [25, 2, 5, 21, 27]. It seems generally understood that a solid formalization is necessary for a sound long-term development of the area. Formal definitions for certain disorder-based primitives have been developed in the abovementioned works, in particular for so-called *"Strong PUFs"*. They are based on the conditional entropy and mutual independence of PUF-responses [5, 21], and/or on the unpredictability of unknown PUF-responses from a set of known responses [25, 2].

However, none of the existing works has developed a *formal machine model* that could *at the same time*:

1. Model physical security features and the physical capabilities of an attacker (e.g., capture the physical unclonability and other physical features of PUFs

and related primitives).

2. Model computational securiy features and the computational capabilities of an attacker (e.g., express the computational security of classical schemes like digital signatures, or express the computational independence of PUF-CRPs).

3. Be employed in formal, mathematical security proofs (e.g., reductionist approaches).

For comparison, the information-theoretic formulation of PUF-security by Brzuska et al. [5], which has been continued by Ostrovsky et al. [21], only addresses the mutual, information-theoretic independence of a PUF's challenge-response pairs (CRPs). It is not built on an underlying physical machine model, and does not formally model the physical unclonability of the PUF. It also does not include physical attacks on the PUF that go beyond mere CRP-measurements. However, the relevance of such extended, physical adversarial methods has just recently been proven by a number of side channel attacks on Arbiter PUFs and variants [18, 32, 33]. The attacks use extra side channel information beyond PUF-CRPs, and break XOR Arbiter PUFs of sizes and complexities that are *not* attackable by mere CRP measurements and subsequent numerical modeling attacks [30]. This shows a gap in the formalism, a gap that is obviously not just theoretically, but also practically relevant. Furthermore, the formal framework of Armknecht et al. [2] does not stipulate a machine model upon which formal security proofs could be based. For this reason their definitions indeed have not been used in any formal security proofs up to this date.

This stresses the need for new, extended definitions in the area. The formal development of such definitions, however, requires a new machine model, one that can incorporate general physical actions of the adversary like side channel measurements.

Our Contributions. We make three main contributions in this work. First of all, we develop a new machine model called *physical Turing machines* (or *PhTMs*, for short). PhTMs are quite similar to standard Turing machines, but have the additional capability to process real physical objects as inputs and outputs. Together with PhTMs, we introduce the concept of a *"technology"*, i.e., of a set of admissible actions upon which the PhTM operates. PhTMs and technologies allow us to model cryptographic parties that use physical approaches in one way or the other.

Secondly, we briefly discuss how Physical Turing Machines can be applied to the formalization of classical, computational complexity based cryptography. We argue that the current formalization in this area, which uses standard Turing machines (TMs), does not include atackers that use inherently physical methods and computations. Examples include adversaries that employ optical or quantum computations in attacking a given encryption or signature scheme. At the same time, we observe that simply replacing standard TMs by quantum TMs in existing definitions will not be satisfactory, since this would overestimate the real-world power of quantum computers, and formally render many schemes unjustifiedly insecure. We argue why shelter can be found in the use of our novel concepts of PhTMs and technologies.

Thirdly and mainly, we exemplarily apply PhTMs to a well-known scheme from the area of disorder-based security and PUFs, lead a full-fledged, detailed reductionist security proof on the basis of PhTMs. The scheme concerns a method for the secure labeling of valuable goods (such as pharmaceuticals, passports, banknotes, etc.) that combines a digital signature with a unique, non-clonable physical object, a so-called *"unique object (UNO)"* [26]. In this context, we first formalize various variants of the notion of a unique, unclonable object, and show relations between the different notions. We then prove that the above scheme is secure under the provisions that the employed object is unique and that the used digital signature scheme is secure. The proof is carried out by a a reductionist technique, and uses PhTMs as the underlying machine model.

One Caveat. We would like to delimit too excessive expectations already at this early point: The aim of this paper is not to clarify in a strict and absolute sense which physical actions are practically possible and which are not. This needs to be resolved by physicists, not mathematicians and computer scientists. As can be seen by the examples of current computational complexity theory and the still unresolved $\mathcal{P}$ vs. $\mathcal{NP}$ question, such a hope would be indeed unreasonably high. Instead, we would like to enable security proofs of the form *"if primitives* A *and* B *are secure, then so is scheme* C*"*, where A, B and C can include both computational *and* physical staments, in opposition to the traditional Turing formalism. As it turns out, this goal of our work is intricate, but also rewarding enough in itself.

Organization of this Manuscript. We take some time in Section 2 in order to prepare the stage for our new machine model and its applications. We review the general purpose of formalizing cryptographic schemes in Section 2.1, and explain a (slightly provoking) conceptual gap in the current, Turing machine based formalization of cryptography in Section 2.2. Then, we will take a brief look at Physical Cryptography in Section 2.3, which illustrates our main motivation for the new, extended machine model. We conclude by a summary in Section 2.4.

The technical contributions of the paper are presented in Sections 3 to 6. In Section 3, we introduce Physical Turing Machines as a formal "computational" model that allows both numeric computations and physical actions (measurement, generation, manipulation, etc. of physical objects). In Section 4, we briefly discuss the application of PhTMs to the formalization of classical, computational complexity based cryptography. In Section 5, we formalize the concept of unique objects (UNOs) in various ways, and lead a few first proofs in order to get used to our formalism. In Section 6, we deal with one of the main applications of unique objects, which is their use as unforgeable "labels" (or markers or tags) for security tokens and goods of value. We formalize the notion of a secure labeling scheme, and prove by a reductionist technique that secure labeling schemes can be constructed from secure digital signature schemes and unique objects. This technical proof is one of the main contributions of the paper. Finally, we conclude the paper in Section 7 by a summary.

2 Background and Motivation

We take the time to gently motivate our approach and PhTMs in this section, and to integrate them into a bigger picture within classical and physical cryptography. We also provide some background on physical cryptography, PUFs, unique objects, and COAs in Section 2.3. The latter has been included with particular hindsight to readers who encounter PUFs for their first time.

2.1 The Purpose of Formalizing Cryptography

In order to motivate and also to contextualize our PhTMs, it is useful to review some of the basics of theoretical cryptography. One central aim of the latter is to formally prove the security of certain cryptographic schemes. We can distinguish between three, not totally disjoint steps related to this task:

1. *Build a mathematical security model.* In this step a mathematical formalism is set up that models the real-world situation in which a given cryptographic scheme is applied.
2. *Conditionally prove the security of cryptographic schemes in the security model.* This step consists of mathematically proving the security properties defined in step 1 under the premise that some additional, unproven assumptions hold.
3. *Uncoditionally prove the security of cryptographic schemes in the security model.* The aim of this step is to prove the security properties expressed in step 1 without making additional, unproven assumptions.

A couple of non-trivial points need to be made. First of all, note that basically any security model is itself subject to implicit and unprovable assumptions. Hence, any result proven in step 2 and 3 is necessarily subject to these assumptions (and, in a strict sense, could never be called unconditional).

Then, note that it is hard to find a formal criterion that distinguishes the assumptions made in step 1 and 2. Associating assumptions to one of the steps seems a matter of human intuition and reasoning, not so much a matter of formal distinguishability. Consider as an example the familiar formalization of complexity based cryptography. As implicit in the security definitions, it is assumed there that the adversary cannot execute any other than polynomially time-bounded Turing computations. This assumption is commonly associated to step 1. The assumption that the adversary cannot factor numbers quickly, however, is regarded as part of step 2. This choice is to some extent arbitrary; from a purely formal perspective, both assumptions could be attributed to the respective other step, too.

Third, it is important to realize that it can be nontrivial to decide whether an unconditional proof of the security properties of some scheme (i.e. step 3) is possible at all in the mathematical formalism provided by step 1. There could be formalisms in which we have to confine ourselves with step 2, because step 3 is generally impossible. Once more, this may hold in particular for the current complexity-based formalization of cryptography: We do not know whether the underlying problem of NP vs. P is independent of the axioms of set theory (ZFC, more specifically); see, for example, [1]. Consequentially we cannot say whether the formal security of many cryptographic schemes, whose unconditional proof would imply that $NP \neq P$, is independent of ZFC. In any case it is obvious that step 3 has *currently* not been completed in the standard formalization of complexity based cryptography.

But — is there any value in a framework which currently or forever confines us to steps 1 and 2? Again, it can be observed by the example of present cryptography that there is – recall that as it stands, the current theory of *complexity based* cryptography is nothing more than a large network of reductions. Its value is nevertheless undisputed within the community.

This justifies to accept mathematical frameworks which initially merely allow for step 2, and in which step 3 seems either very remote or even problematic in principle. We will come back to this conclusion in Section 3 in order to justify our new Turing model. For now, let us turn to one particular aspect of the standard mathematical framework of complexity based cryptography in the next section.

2.2 The Turing Machine and the Foundations of Cryptography

The Turing machine formalism has traditionally been used as the foundation of computability and complexity theory, and consequently also in the formalization of complexity-based cryptography. This makes two implicit assumptions, as pointed out in [36]. First of all, both computability theory and complexity theory often implicitly assume the validity of the well-known Church-Turing Thesis (CT) [36]. Secondly, complexity theory in large parts also assumes the (perhaps less well known) Extended Church-Turing Thesis (ECT), which can be formulated like this [36]:

> **Extended Church-Turing Thesis (ECT):** Any function on the naturals that is computable *efficiently* (i.e. in polynomial time) in some intuitively reasonable sense can also be computed *efficiently* (i.e. in polynomial time) in the Turing machine formalism. In particular, any function that can be computed in polynomial time by some physical hardware system can be computed in polynomial time by a Turing machine.

As complexity theory and the Turing machine formalism are the main tools in the formalization of complexity based cryptography, CT and ECT are implicit, but often overlooked assumptions in that area. Quite unfortunately, there is some evidence that ECT could be false, as factoring can be done in polynomial time on a quantum computer, but, many suspect, not on a Turing machine. This makes the following future scenario possible: While we can prove *unconditionally* that breaking RSA cannot be done in polynomial time on a Turing machine, we can at the same time factor efficiently in practice by the use of quantum computers. RSA would then be unconditionally and provably secure in theory according to our current formalization standards, while it was obviously insecure in practice. This seems to indicate that the current foundations of cryptography exhibit a conceptual gap when it comes to computations that are executed by real world physical systems and not by Turing machines; the attempt to close this gap seems reasonable.

One obvious, very rigid approach countermeasure would be to substitute quantum Turing machines instead of ordinary TMs in all definitions and proofs. Unfortunately, not many popular asymmetric cryptographic schemes are left secure when we make that step, and most proofs break down. Also, it seems far away from practice, as at the moment quantum computers are just able to factor low-range two digit numbers.

Do we have other, more realistic alternatives? It seems that our only option was to ignore the gap, adopting the position that it is *practically irrelevant*. However, lack of practicality is an argument that at times has been used unrightfully against the foundations of cryptography as a whole; it does not feel appropriate to turn it the other way, and to use it against a correct objection to the current standards within the foundations of cryptography. We believe that there is no use in pursuing the foundations of cryptography half-heartedly; if we take the soundness of cryptography serious, if we really are to *"build a long-lasting building and not a cabin"*, then we should include quantum attacks into our model.

Still, this leaves us with the question how this can be done. Ideally we would like to set up a computational model similar to the Turing formalism, in which we can express and prove security properties at least conditionally, possibly even unconditionally. This formalism should include quantum and other physical computations in order to avoid the conceptual gap described earlier. On the other hand, it must not allow quantum attacks way beyond current technology, as otherwise many asymmetric cryptographic schemes of interest become *unrealistically insecure* in our model. The only way out of this dilemma seems a formalism that can somehow include the current state of technology, while still enabling (at least conditional) security proofs.

2.3 Physical Cryptography

Besides the potential gap in the current formalization of classical cryptography that we addressed in the last section, there is a second, and perhaps yet more important, motivation for the introduction of physical Turing machines. This motivation is is related to some recent developments in security, in which disordered, practically unclonable physical systems are used in cryptography and security. The probably best known concepts from this area are physical unclonable functions (PUFs) [22, 12] and certificates of authenticity (COAs) [10], but there are also others; a good overview is given in [26]. As mentioned in the introduction, the corresponding research area could be (and has been) termed *"physical cryptography"* [28] or *"disorder-based security"* [26].

The explicit use of physical structures in cryptography and security makes PhTMs useful in at least two respects: Firstly, in the formal definition of those security features that are required from PUFs, for example, in a certain application. Secondly, in formal proofs that show the security of, for example, a given PUF-based or COA-based scheme under certain assumptions.

In order to achieve a relatively self-contained treatment, we will familiarize readers with two example schemes from physical cryptography, one of them relating to COAs, the other to PUFs. The former scheme, which is based on so-called "unique objects"

[26] plays a key role for our paper, as it will be treated and proven formally in Sections 5 and 6.

2.3.1 Unique Objects and Certificates of Authenticity

Following [26, 10], unique objects (UNOs) are physical systems or objects that exhibit a set of analog properties which cannot be copied, reproduced, or manufactured by intent. These properties should be detectable reliably by some external measurement apparatus, and they should be expressible in a relatively short binary string (a rule of thumb would be a size below 10 kB [26]). Even if the analog properties and the details of the measurement apparatus are given to an adversary, he shall be unable to fabricate a second object that exhibits the same properties upon measurement with the apparatus. Under these circumstances, we also call said properties the unique properties of the unique object.

Unique properties often occur due to uncontrollable variations in the manufacturing process. One easy conceptual example of a unique object is a random distribution of (possibly only a few) optically active particles. Such a distribution is hard to reproduce or to produce on intent. The unique properties, which could be measured by an external measurement apparatus, could consist of the individual position of each single particle. [1] A second example is a paper surface, in which the so-called paper fibers take random, interwoven, three-dimensional positions (compare [6]).

Let us now consider one typical application of unique objects: The generation of unforgeable and machine-testable labels (tags/markers) for any products or goods of value.

Application: Labeling of valuable objects. To label products unforgeably is a problem of high theoretical appeal and also of economic relevance. Even Newton reportedly dealt with the generation of forgery-proof coins in his position of the Master of the Mint in the early 18th century. Nowadays, it has been estimated that the worldwide economical damage caused by faked brand products amounts to several hundred billion Dollars per year [14]. The basic task can be described as follows: Given a valuable product, generate a physical token – the label – that can be applied to the product such that the following conditions are met:

1. The validity of the label can be tested by an automatized device.
2. The label cannot be faked or counterfeited.

Unique systems suggest themselves as unforgeable labels, as they have properties that cannot be copied or reproduced. However, the propery of being unreproduceable also leads to problems: All labels that are applied to different specimen of the same product differ and are subject to random production variations. How shall the testing device

[1] Please note that this is in opposition to the approach of optical PUFs of Pappu et al. [22], where the measured speckle pattern is a function of the positions of all particles, and does not directly measure the position of individual, single particles.

distinguish a 'random' label that has been produced by the legitimate company from a 'random' label produced by a fraudster? The idea is to use a standard technique from mathematical cryptography, namely digital signatures, in connection with unique objects. The combined labeling scheme works as follows:

1. Produce a (random) unique object, and measure its unique properties $P_1, \ldots, P_n$.
2. Create a digital signature $S \overset{\text{def}}{=} Sig_K(P_1, \ldots, P_n)$ for these properties by use of some secret key K.
3. Apply the (numerical) signature, the (numerical) description of the properties $P_1, \ldots, P_n$ and the (physical) unique object to the product.
4. All testing devices are equipped with the public verification key P that corresponds to K. If a labeled product is inserted into some testing device, it executes the following procedure:

    (a) Check if the signature S is a valid signature for the properties $P_1, \ldots, P_n$ listed on the product. To that end, use the public verification key P.

    (b) Test if the unique physical system contained on the product has the properties $P_1, \ldots, P_n$ listed on the product.

    If this is the case, the testing device regards the label as valid, otherwise as faked.

Intuitively, this labeling technique seems secure provided that the physical system really is unique and that the digital signature scheme is secure. But — can we prove that? How could we set up a formal framework in which such a proof can be conducted?

The difficulty of this task lies in the fact that attacks on the labeling scheme can be executed on two levels: First of all on a binary level by faking the digital signature. Another possibility, however, is to attack the scheme on a physical level by trying to copy the unique physical system. Therefore modelling the attacker as a standard Turing machine will not suffice. Instead, we should use a machine model which combines the ability for Turing computation with the capability to process physical objects; this model could have some 'Turing part' and some 'physical part'. Again, the capabilities of the physical part should realistically operate only within the limits of current technology, which is a condition that we have encountered before (Section 2.2), but do not know how to meet yet.

2.3.2 Physical Unclonable Functions

A physical unclonable function (PUF) is a physical system S which possesses a certain level of disorder or randomness in its micro- or nanoscale structure. S can be excited with so-called external stimuli or challenges C_i, upon which it reacts with corresponding responses R_{C_i}. These must be a function of the applied challenge and of the structural disorder that is present in the PUF. The responses are supposed to be stable over time and multiple measurement, and the tuples (C_i, R_{C_i}) are commonly called the *challenge-response pairs (CRPs)* of the PUF (see [22, 12, 26]).

It is usually assumed that a PUF cannot be cloned or reproduced exactly, not even by its original manufacturer. This well established assumption is not based on a fundamental physical theorem, such as the no cloning theorem in quantum mechanics. Instead, it is viable in practice due to the inherent limitations of current nanofabrication techniques. These are unable to position molecules or atoms with arbitrary precision in three dimensions [22], and hence cannot reproduce the small-scale disorder and structural randomness of the PUF exactly.

So-called Strong PUFs have a second important property: They allow a very large number of applicable challenges and possess a complex, inimitable challenge-response behavior. It is assumed that their responses cannot be predicted numerically, but can only be obtained by a physical measurement on the unique and unclonable PUF itself. This must hold even if an adversary had access to the PUF at earlier points in time, could freely apply challenges to the PUFs, and could measure the corresponding responses. In other words, even if a large number of challenge-response pairs of a Strong PUF are known, the challenge-response behavior cannot be machine learned or modelled well enough to allow the certain numerical prediction of the responses to new, randomly chosen challenges. This property could be referred to as the *unpredictability* or *non-learnability* of a Strong PUF [26, 25, 31].

In a nutshell, the difference between unique objects and Strong PUFs lies in the large number of CRPs a Strong PUF allows, and in the fact that the measurement signal of a unique object is analog and determined by an external apparatus, while the CRPs of a Strong PUF *may* be digital and *may* be determined by an integrated measurement apparatus. Furthermore, a unique object must remain secure even if its the unique properties are given to the adversary. More details can be found in [26].

Application: Secret Key Exchange by Physical Unclonable Functions (PUFs). (Strong) PUFs are a very powerful cryptographic tool: They allow identification, key exchange, oblivious transfer and other applications [24, 5, 21]. In the following, we sketch a variant of a key exchange protocol on the basis of Strong PUFs.

We assume that:

1. Alice holds a genuine Strong PUF S at the beginning of the protocol, which has been fabricated by herself or by a trusted manufacturer.

2. Alice and Bob have an authenticated (but non-confidential) binary channel and a fully insecure physical channel at hand.

3. The protocol is executed only once, and no one has got access to the Strong PUF anymore after the end of the protocol.

Our focus thereby is not on describing all formal details such as error correction (compare [5]), but on providing the reader with the basic idea of the scheme. The explicit step for authenticatiog the sent PUF (Step 5 in the protocol) is new compared to existing schemes [5].

The key exchange scheme works as follows:

1. Alice chooses random challenges $C_1, \ldots, C_{2k}$. She measures the PUF S in order to determine the responses $R_{C_1}, \ldots, R_{C_{2k}}$.

2. Alice sends the Strong PUF S over the physical channel to Bob.

3. Bob receives an object S', which is not necessarily equal to S (recall that it could have been exchanged by the adversary, since the physica channel is insecure).

4. Bob sends the message "I got an object!" over the authenticated binary channel to Alice.

5. Alice and Bob check that S is equal to S'. That is, they check that Bob received the object that was sent away by Alice, and that the object has not been exchanged or manipulated while it was sent. To that aim, they execute the following subprotocol:

   (a) Alice sends the values $C_1, \ldots, C_k$ to Bob.

   (b) Bob measures the object S' with the parameters $C_1, \ldots, C_k$ and receives the values $V_1, \ldots, V_k$, which he sends to Alice.

   (c) Alice checks if the values she got from Bob match the values she measured herself in step 1. That is, she checks if $V_i = R_{C_i}$ for $i = 1, \ldots, k$. If this is the case, she sends the message "Ok." over the binary channel to Bob. Otherwise, she sends "Stop!" over the binary channel, and Alice and Bob abort the protocol.

6. Alice sends the values $C_{k+1}, \ldots, C_{2k}$ over the binary channel to Bob.

7. Bob determines the values $R_{C_{k+1}}, \ldots, R_{C_{2k}}$ by measurement on the PUF.

8. Alice and Bob extract their secret binary key from the values $R_{C_{k+1}}, \ldots, R_{C_{2k}}$, which is known to both of them.

Why is the above scheme secure? We can argue informally in favor of its security as follows. By the properties of the (Strong) PUF, an adversary Eve who might have access to the PUF while it is delivered physically from Alice to Bob cannot fully read out all CRPs, cannot clone the PUF, and cannot build a numerical simulation model of the PUF. Hence, the probability that Eve by chance reads out a piece of information of the system that is later used to build the secret key is very small; using privacy amplification techniques when extracting the key from the bit sequence $R_{C_{k+1}}, \ldots, R_{C_{2k}}$ can make it arbitrarily small.

Again this leaves us with the question how can we prove the security properties in a *formal* way. Apparently, an adversarial model designed for that purpose once more has to include both capabilities for binary information processing and for physical attacks.

These physical attacks may include attempts to copy, photograph, scan, imprint the PUF into a suitable material to form a negative copy of it, or to otherwise physically process the PUF. Therefore a standard Turing machine again will not suffice as a model for the attacker. Also an oracle Turing machine which models the PUF by an oracle that can be presented with a challenge C and returns the value R_C will not do. The same holds for mere information-theoretical Strong PUF definitions as the ones presented in [5]. All of these formal approaches cannot model general physical attacks such as those listed above.

Hence, just as in the case of the labeling scheme, the model for formalizing PUFs must include capabilities for Turing computation as well as for the general processing of physical objects. In any realistic model those physical capabilities must be subject to the state of technology, an expression beginning to sound familiar.

2.4 Summary

We have covered a range of topics in the introduction that spans from the purpose of formalizing cryptography to new approaches in cryptography. The following conclusions, which will be relevant in the upcoming parts, can be drawn from the presented material:

1. The purpose of formalization in cryptography does not only lie in enabling unconditional security proofs. Formal reductions among cryptographic notions and conditional security proofs are other sufficient goals. Hence, models which do not allow for unconditional proofs now or in principle should not be discarded for that fact.

2. We discussed a conceptual gap in the current foundations of cryptography that should ideally be closed. This gap relates to physical attacks in general, and more specifically to (i) physical computations like quantum computations that might eventually outperform classical computers, and (ii) to physical attacks on cryptographic hardware like invasive attacks or side channels. Closing the gap could only be done by introducing a machine model which combines the capabilities for Turing and for physical computation and which also models physical actions on hardware systems. Furthermore, we observed that it seemed reasonable to limit the physical computations in that model by the state of current technology (for example when it comes to including quantum computers in the picture), but we did not know how to achieve this yet.

3. We presented some new developments in cryptography, including PUFs, COAs and other structures, which we subsumed under the name *"physical cryptography"* or *"disorder-based security"*. These approaches use the analog physical properties of disordered physical systems for cryptographic purposes. A strict formalization of this area would make it necessary to add the ability to process physical objects to the capabilities of standard Turing machines. Current formalization attempts in the area have not yet addressed this problem.

The above conclusions motivate the following tasks of research: First, introduce a new machine model that includes the potential for physical actions on physical objects, and a way to formalize the notion of "current technology". Second, reformulate the security properties of standard binary cryptoschemes and also the corresponding security proofs in that new model, thereby addressing the potential gap in the current formalization of cryptography. Third, define the security-relevant properties of physical systems that are applied in physical cryptography in the new model. Fourth, conditionally prove the security properties of schemes of physical cryptography in the new security model. The aim of this paper is to cover some aspects of this new line of research.

3 Physical Turing Machines

3.1 Informal Description

This section is devoted to an informal description of the physical Turing machine model; a more formal presentation will be given in section 3.2. Put in one sentence, the aim of physical Turing machines is to model computations that are executed by human beings with the help of physical systems. We imagine the situation as follows:

The human being holds paper and pencil in order to make some private notes or private computations. He has a finite number of finite physical systems or machines $S_1, \ldots, S_n$ under his control, which he can let perform computational tasks for him. These computational tasks can take numbers and/or physical objects as input, and produce numbers and/or physical objects as output. In order to enable information exchange between the systems and the human being, we envision that the each system has got a digital interface, into which the human being can type information, and through which the human being can receive information.

Whenever the human being wants one physical system to perform a certain numerical computation, he types the numerical input of that computation into the interface. After some computation time, he gets the numerical result of that computation, communicated over the interface.

Whenever the human being wants one physical system to perform some action on one or more given physical objects, we imagine that the human being presents the objects to the system by placing them in some specific position relative to the system. These positions also impose an order on the input objects. Then, the human being uses the interface in order to provide the physical system with some accompanying numerical input. That input can, for example, describe how the system should process the object, but is not limited to that purpose. After some computation time, the human being gets in return one numerical string communicated by the interface, and possibly one or more processed objects, which are placed in some specific positions relative to the system. Again, positioning places an order on the returned objects.

We assume that a human equipped as described can perform certain tasks that we subsume under the term computation. The tasks to be performed are presented to him in the form of an *input*, consisting of a binary number and/or a finite number of physical objects. The solution to the task is presented by the human being as a certain *output*, again consisting of a binary number and/or a finite number of physical objects. Hence, the actions of the human being can be seen as the computation of a function F, whose domain and range are the set of finite tuples that consist of a finite binary string and a finite number of physical objects. As computation time we can naturally regard the time interval experienced by the human being between the two events of being presented with the input and presenting the output.

That rough model seems generally fine, but one aspect needs further consideration. As the physical systems $S_1, \ldots, S_n$ employed by the human being during a computation are finite, they can in general only deal with a finite range of inputs and outputs. We would, however, like to be able to compute infinite functions in our formalism. Hence, the physical systems $S_1, \ldots, S_n$ in general should be able to deal with an unrestricted input and output range. This is a contradiction and raises a problem.

In the informal setting just described, our (idealized) human being could practically encounter this problem by building the physical systems that support his computation bigger and bigger, adjusting to the growing size of the input. We will model this behavior as follows: Each of the above 'machines' $S_1, \ldots, S_n$ is represented in our model by an infinite sequence of machines,

$$S_i = S_i^1, S_i^2, S_i^3, \ldots, \quad \text{for } i = 1, \ldots, n.$$

There, each single machine S_i^j is required to have finite mass, but within any sequence S_i the masses of the single machines may grow beyond any threshhold to adjust to more complex inputs.

In any one given physical Turing machine computation, then, the physical Turing machine is allowed to use precisely one machine from each infinite sequence S_i. These n machines must be selected deterministically by a previously specified choice function, which may only depend on the length of the binary input and the weigth of the physical input. This mechanism is reminiscent of the choice mechanism in the computational model of polynomial size circuits. It can be formalised conveniently as specified in Definition 3.5, after a norm on mixed numerical/physical inputs has been introduced.

In any case, the described choice mechanism forces the physical Turing machine to merely use a finite number of computing machines (and not the whole infinite sequences S_i) in each computation, which is desirable.

3.2 Definition of Physical Turing Machines

The informal discussion of the previous section gives rise to the following sequence of formal definitions, which eventually lead to the notion of a physical Turing machine. We start by defining the physical stage on which physical Turing machines live, which we call a 'universe'.

Definition 3.1 (Deterministic Universes)**.** *A (deterministic) universe $\mathcal{U}$ is a 4-tuple $\mathcal{U} = (\mathcal{O}, \mathcal{M}, \mathcal{F}, m)$ with the following properties:*

1. *$\mathcal{O}$ is an arbitrary set called the set of possible objects.*

2. *$\mathcal{M}$ is an arbitrary subset of $\mathcal{O}$ called the set of possible machines.*

3. *$m : \mathcal{O} \to \mathbb{N}$ is a mapping called the mass.*

4. *$\mathcal{F} : \mathcal{M} \longrightarrow \mathsf{Func}\big(\{0,1\}^* \times \mathcal{O}^* \to \{0,1\}^* \times \mathcal{O}^* \times \mathbb{N}\big)$ is a mapping called the behavior function of $\mathcal{M}$.*

5. *Both $\mathcal{O}$ and $\mathcal{M}$ contain a distinguished object $\lambda_{\mathcal{O}}$ called the emtpy object, for which it holds that $m(\lambda_{\mathcal{O}}) = 0$.*

Definition 3.2 (Notational Conventions for Universes)**.** *We make the following notational conventions:*

1. *The set $\mathcal{O}^*$ denotes the set of finite tupels of elements of $\mathcal{O}$. Whenever we refer to an element X of some set $S \times \mathcal{O}^*$, we often write $X = (s, O_1, \dots, O_n)$ instead of $X = (s, (O_1, \dots, O_n))$.*

2. *For every $M \in \mathcal{M}$, $\mathcal{F}(M)$ is a mapping, which we often denote by $\mathcal{F}_M$. That mapping can be split up in three components $\mathcal{F}_M = (\mathcal{F}_M^1, \mathcal{F}_M^2, \mathcal{F}_M^3)$, where $\mathcal{F}_M^i \overset{\text{def}}{=} \Pi^i(\mathcal{F}_M)$ with Π^i denoting the projection onto the i-th coordinate.*

Before we finally define physical Turing machines, we have to sort out two more formal details.

Definition 3.3 (Physical and Binary Tapes and Their Content)**.** *Let $\mathcal{U} = (\mathcal{O}, \mathcal{M}, \mathcal{F}, m)$ be a universe. A physical tape (in $\mathcal{U}$) is a half-sided infinite tape whose cells can contain any object from the set $\mathcal{O}$. A physical tape is said to be empty if all cells contain the empty object. Further, its cells are labeled by natural numbers in increasing order, starting with the number 'one' at the end of the tape. The content of a physical tape is a (finite or infinite) sequence $O_1, O_2, \dots$ where the objects O_i of the sequence are the objects contained in the cells of the tape, in increasing order of the labels of the cells, omitting all empty objects contained in the cells. If the content is a finite sequence, we can write it as tuple $(O_1, \dots, O_k)$.*
A binary tape (in $\mathcal{U}$) is a standard, half-sided binary Turing tape, whose cells can contain the symbols zero and one and the symbol 'blank'. The tape is empty when all its cells contain the blank symbol. The content of a binary tape is the finite or infite sequence of bits contained in the cells, omitting blanks. If that sequence is finite, we can write it as a finite binary string x.

As the input and output of a PhTM are a mixture of numbers and physical objects, we should define a norm on such inputs.

Definition 3.4 (Input and Output Norm)**.** *Let $\mathcal{U} = (\mathcal{O}, \mathcal{M}, \mathcal{F}, m)$ be a universe. We define a norm $\|\cdot\|$ on all elements $X = (x, (O_1, \dots, O_n))$ of the set $\{0,1\}^* \times \mathcal{O}^*$ by*

$$\|X\| \overset{\text{def}}{=} \max\{length(x), \Sigma_{i=1}^{k} m(O_i)\}$$

where $length(\cdot)$ denotes the length of a binary string.

Now we are in a position to define physical Turing machines.

Definition 3.5 (Physical Turing Machines)**.** *Let $\mathcal{U} = (\mathcal{O}, \mathcal{M}, \mathcal{F}, m)$ be a universe. A physical Turing machine or φ-Turing machine M in $\mathcal{U}$ consists of a Turing programm P together with n infinite sequences $M_1, \ldots, M_n$ of machines in $\mathcal{U}$, where $M_i = (M_i^0, M_i^1, M_i^2, \ldots)$ for $i = 1, \ldots, n$.*
M has got five tapes: Two internal tapes, two external tapes, and one switching tape. One of the internal tapes is a binary and the other one is a physical tape; the same holds for the external tapes; the switching tape is binary. Further, a PhTM has got a counter, which cannot be accessed or deliberately altered by the PhTM during computation. The counter shows the value 0 at the beginning of each computation.
The internal tapes and the switching tape are initially empty. The content x of the binary external tape and the content $(O_1, \ldots, O_n)$ of the physical external tape at the beginning of the computation are called the binary respectively physical input of M; the string $X \overset{\text{def}}{=} (x, O_1, \ldots, O_n)$ is the (overall) input of M.

M further has got $n+1$ distinguished states termed the calling states Call 1, Call 2, ..., Call n and the swapping state. Suppose that when M switches into a calling state Call i, the overall input of the computation was X, and that at the time of switching the content of the binary external tape was x and the content of the physical external tape was $(O_1, \ldots, O_k)$. Then, the following happens within one Turing step:

1. *The content of the binary external tape is erased and replaced by*
$$\mathcal{F}^1_{M_i^{\|X\|}}(x, O_1, \ldots, O_k).$$
2. *The content of the physical external tape is erased and replaced by*
$$\mathcal{F}^2_{M_i^{\|X\|}}(x, O_1, \ldots, O_k).$$
3. *The counter of the PhTM is increased by the value*
$$\mathcal{F}^3_{M_i^{\|X\|}}(x, O_1, \ldots, O_k).$$

Likewise, we describe what happens when the physical Turing machine switches into the swapping state. We distinguish between two cases:

1. *When M switches into the swapping state, the content of the swapping tape is of the form (x, y), where x and y are natural numbers (assuming a previously specified encoding scheme). Then, the content of cell x of the external physical tape is exchanged with the content of cell y of the internal physical tape, and the swapping tape is erased. This takes one Turing step.*
2. *The content of the swapping tape is not of that form. Then nothing happens. Again, this takes one Turing step.*

The output of a Turing machine is the content of the external tapes when it switches into an end state, with the numerical output being the content of the binary tape, and the physical output being the content of the physical tape. The computation time of a Turing machine is the number of Turing steps plus the content of the counter of the Turing machine when the Turing machine switches into an end state.

Before we can briefly discuss the definition of a PhTM in the next subsection, there are a few more things that need to be said. First of all, we will introduce a formal notation for the output of a PhTM.

Notation 3.6 (Output of physical Turing machines)**.** *Let M be a PhTM and $X \in \{0,1\}^* \times \mathcal{O}^*$. The output of M on input X is denoted by $Out_M(X)$. Unless otherwise stated, we often write $M(X)$ instead of $Out_M(X)$.*

The next definition states what it means for a PhTM to be ressource efficient; as usual, ressource efficiency is taken as being polynomially bounded in some way.

Definition 3.7 (Polynomial PhTMs)**.** *Let $\mathcal{U}$ be a universe, and let $M = (P, M_1, \ldots, M_n)$ be a PhTM of $\mathcal{U}$. The we define the following notions:*

1. *M is called a polynomial time φ-TM if there exists a polynomial p such that for any $X \in \{0,1\}^* \times \mathcal{O}^*$, the computation time of M on input X is less or equal to $p(\|X\|)$.*
2. *M is called a polynomial mass PhTM is there exists a polynomial p such that $m(M_i^k) \leq p(k)$ for all natural numbers k and $1 \leq i \leq n$.*
3. *M is called a polynomial PhTM if is both a polynomial time and a polynomial mass PhTM.*

Finally, a computation executed by a PhTM M is called a polynomial time or polynomial mass or polynomial computation if M is a polynomial time or polynomial mass or polynomial PhTM, respectively.

Now we can state an important feature of PhTMs, namely that they can be customized with respect to a certain 'state of knowledge' or 'state of technology' in a universe. Intuitively, a state of technology should be given by the machines that can (in practice, not in principle) be built in a certain universe; hence it is suggestive to consider a state of technology to be a subset of the set of all possible machines. This motivates the next definition.

Definition 3.8 (State of Technology)**.** *Let $\mathcal{U} = (\mathcal{O}, \mathcal{M}, \mathcal{F}, m)$ be a universe. Any set $\mathcal{T}$ that is a subset of $\mathcal{M}$ is called a state of technology (in $\mathcal{U}$); whenever the universe is clear from the context, we drop it. If a machine M is contained in $\mathcal{T}$, we also say that M is a machine of $\mathcal{T}$.*

We can now say what it means that a PhTM respects a state of technology.

Definition 3.9 (PhTMs Respecting a State of Technology). *Let $\mathcal{U}$ be a universe, and $\mathcal{T}$ be a state of technology in $\mathcal{U}$. Let $M = (P, M_1, ..., M_n)$ be a PhTM in $\mathcal{U}$. We say that M respects $\mathcal{T}$, or that M is a PhTM in $\mathcal{T}$, if all elements of the sequences M_i are machines of $\mathcal{T}$.*

This, for the moment, completes the definition of the concepts that we had been asking for: We have formally defined what a physical Turing machine is, we have found a way to express the informal phrase 'state of technology', and we have defined how a state of technology may influence the power of a Turing machine. Our definitions will be discussed briefly in the next subsection.

3.3 Discussion of Physical Turing Machines

There is one obvious objection that can be raised against physical Turing machines: Their definition does not specify *which* machines respectively *which* functions can be employed to support the standard Turing machine computation. Hence, their precise computational power remains unclear and seems basically undefined. There are various ways to encounter this objection.

First of all, we remark that PhTMs were introduced in order to reformulate complexity based cryptography and to provide a formal basis for physical cryptography, among other reasons. As discussed at length in section 2.1, it suffices for that purpose if PhTMs can be used for the following purposes: First, to relate the security of different cryptographic schemes and primitives to each other; second, to lead *conditional* security proofs. To that aim no precise definition of the power of the computational model is necessary, as we will see in detail in the upcoming sections 6.3. Rather, a framework with the following properties is needed: First, it can be used to express certain assumptions about the security of cryptographic schemes; second, it is general enough to capture all conceivable attacks; finally, it has enough internal structure to allow reductions between different cryptographic objects. These properties are met by PhTMs, and, when it comes to the second aspect, even more than by standard TMs.

Second, there is no reason why *in principle* and *in practice* it should not be possible to fully determine the set of physical systems and corresponding functions that can be employed in a PhTM computation. It is well conceivable that as the state of knowledge in theoretical physics progresses, the notion of a physical system can be formalized well enough in order to fully determine the behavior of physical systems. Together with specifying a decent input - output formalism, this might lead to a full characterization of the finite functions that can be computed by finite physical systems according to the currently most advanced physical theory. More precisely, it is conceivable to apply the most advanced physical theory (which today would be quantum or string theory) to the physical parts of any PhTM in order to restrict

(A) the set of possible objects and possible machines in the universe $\mathcal{U}$, from which a PhTM might draw its physical parts.

(B) the technology $\mathcal{T}$ from which a PhTM might draw its physical parts.

That means that there is some good reason to assume that at least *potentially* and *in principle* we can fully determine the computational power of PhTMs, or of polynomial PhTMs, as our state of knowledge progresses. Please note that nothing more can be said, in fact, about the status of classical TM or classical polynomial-time TM computations. Deciding whether a function is Turing computable is not Turing computable; further, we do not know how to resolve the NP vs P question, left alone that we cannot say whether this question can *in principle* be decided in ZFC. This is not held against the Turing formalism, and we feel it should also not be held against our formalism.

One other important aspect follows: Had we applied one specific physical theory in the definition of PhTMs, then this theory might be subject to change or even replacement over time. The general formalism provided by PhTMs, however, can persist independent of such change: We simply determine the universes $\mathcal{U}$ in which the PhTMs operate, or the technologies $\mathcal{T}$ they respect, by application of the new physical theory instead of the old one. This seems to further stress the generality of our approach to physical computation: PhTMs can persist while physical theories change.

Fourth, one interesting historical reference ought to be made in this context. When he introduced the Turing machine (respectively LCM, as called then), Turing's aim was obviously not to introduce a complete computational model that covers any computation executable by a *physical* machine. Rather, he intended to formalize any 'mechanical' procedure that could be carried out by a *human being.* This approach was motivated by his goal to apply the Turing machine to Hilbert's Entscheidungsproblem, which asked for a *humanly executable* procedure of a certain, 'mechanical' sort, where 'mechanical' has some specific meaning that deviates from our everyday language. Turing was to show that there was no such procedure in the case of predicate logic, and the Turing machine was tailored for just this case. Indeed, Turing often spoke of the Turing machine as of a "human computer", or stated that "a man provided with paper, pencil and rubber, and subject to strict discipline, is in effect a universal Turing machine". Perhaps even more strikingly to the point, Wittgenstein wrote: "Turing's 'Machines'. These machines are *humans* who calculate." The approach we took seems to fit well with these statements: First of all, they confirm that it seems necessary to specifically introduce physical systems if we want to go beyond the human-oriented power of the Turing machine. Second, given Turing's and Wittgenstein's views, it seems suggestive to model a human being utilizing physical systems for computation as a Turing machine with access to physical systems, just as we did.

Fifth, we would like to comment briefly on the related notion of oracle Turing machines. Our physical Turing machines can be seen as similar to oracle Turing machines that draw their oracles from a predetermined set of functions. There are some differences, however: First of all, standard oracle machines do not operate on physical objects. Second, standard oracle machines have their oracle fixed for all inputs and do not choose the computational power of the oracle in dependence of the input length. This property of physical Turing machines rather is reminiscent of computational circuits. Third, there is no additional component in the answer of a standard oracle that could be taken as a counterpart to the physical computation time that is added on the counter of a PhTM. Obviously, these differences could have been overcome by adapting the oracle machine framework, but we felt that this might have overstretched this concept a little. Nevertheless, similarities remain, and – another historical remark – there is some resemblence also between our approach and Turing's O-machines, introduced in chapter four of his PhD-Thesis.

Finally, we would like to conclude by once more stressing the necessity to include physical aspects in any computational model that claims to be complete in some sense. One purely mathematically motivated definition alone cannot do, as computation in the end is something physical, not mathematical. In particular, any model used as a fundament in cryptography should be able to include any efficient physical computations, which is what our model attempts to perform.

3.4 Probabilistic and Oracle Physical Turing Machines

We will introduce two variants of PhTMs in this subsection: Probabilistic physical Turing machines and oracle physical Turing machines. Both will be important for formulating concepts in cryptography and physical cryptography; in fact, physical probabilistic Turing machines will be more important for that purpose than (standard) physical Turing machines. Oracle physical Turing machines are probabilistic PhTMs equipped with an oracle; they will, for example, play a role in definition of the physical security of digital signature schemes.

Formulating the concept of physical probabilistic Turing machines makes it necessary to quickly review some concepts from probability theory, and to introduce the concept of a probabilistic universe.

Definition 3.10 (σ -Algebras)**.** *Let Ω be an arbitrary set. A family $\mathcal{A}$ of subsets of Ω is called a σ-Algebra in Ω, if the following holds:*

1. $\Omega \in \mathcal{A}$.
2. *If $A \in \mathcal{A}$, then $A^c \in \mathcal{A}$.*
3. *If $A_1, A_2, A_3, \ldots \in \mathcal{A}$, then $\bigcup_{i=1}^{\infty} A_i \in \mathcal{A}$.*

Notation 3.11. *It can be shown that for any given subset S of Ω there is a smallest σ-algebra in Ω which contains S. This σ-algebra will be denoted by $\mathcal{A}(S)$ in the sequel.*

Definition 3.12 (Probability Spaces)**.** *A probability space $\mathcal{P}$ is a triple $\mathcal{P} = (\Omega, \mathcal{A}, P)$ consisting of a non-empty set Ω, a σ-algebra $\mathcal{A}$ in Ω and a mapping $P : \mathcal{A} \to [0, 1]$ called the probability measure. P has the following properties:*

1. *$P(A) \geq 0$ for all $A \in \mathcal{A}$.*

2. *$P(\Omega) = 1$.*

3. *For all disjoint $A_1, A_2, \ldots \in \mathcal{A}$ it holds that*

$$P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i).$$

Definition 3.13 (Random Variables in Probability Spaces)**.** *Let $\mathcal{P} = (\Omega, \mathcal{A}, P)$ be a probability space, and let $\mathcal{P}' = (\Omega', \mathcal{A}')$ be a tuple consisting of an arbitrary set Ω' and a σ-algebra $\mathcal{A}'$ in Ω' (such a tuple is commonly referred to as 'measurable space'). A mapping $f : \Omega \to \Omega'$ is called a random variable if*

$$f^{-1}(A') \in \mathcal{A} \quad \text{for all } A' \in \mathcal{A}'.$$

Please note that the corresponding definitions for discrete probability spaces and random variables are much simpler; however, we cannot assume that the object set on which the PhTMs operate is countable. This enforces the more general treatment.

We can now 'randomize' our universes, allowing physical processes to be probabilistic. This change will be reflected through altering the behavior function: It will no more map physical systems M to a function f_M, but to a probability space $(\Omega, \mathcal{A}, P_M)$. These probability space have in common that the sets Ω and the σ-algebras $\mathcal{A}$ are the same for any two physical system M_1 and M_2 in $\mathcal{M}$. Only the probability measure $P_M : \mathcal{A} \to [0, 1]$ varies in dependance of M.

Definition 3.14 (Probabilistic Universes)**.** *A probabilistic universe $\mathcal{U}$ is a 4-tuple $\mathcal{U} = (\mathcal{O}, \mathcal{M}, \mathcal{F}, m)$ with the following properties: $\mathcal{O}$ is an arbitrary set called the set of possible objects, $\mathcal{M}$ is an arbitrary subset of $\mathcal{O}$ called the set of possible machines and $m : \mathcal{O} \to \mathbb{N}$ is a mapping called the mass. Further, $\mathcal{F}$ is a mapping called the behavior function, which has the following property:*

$$\mathcal{F} : \mathcal{M} \longrightarrow \left\{ (\Omega, \mathcal{A}, P_M) \;\middle|\; \begin{array}{l} \Omega \;=\; \{0,1\}^* \times \mathcal{O}^* \times \{0,1\}^* \times \mathcal{O}^* \times \mathbb{N} \\ \mathcal{A} \;=\; \mathcal{A}\big(\{F \mid F \text{ is a finite subset of } \Omega\}\big) \\ (\Omega, \mathcal{A}, P_M) \text{ is a probability space} \end{array} \right\}$$

Notation 3.15. *The probability space that is associated with a machine $M \in \mathcal{M}$ is often termed $\mathcal{P}_M$ instead of $\mathcal{F}(M)$. Further, as already implicit in the above definition, the probility measure of that probability space will be referred to as P_M.*

We can now use the probability space $\mathcal{P}_M$ in order to define the output and the computation time of M as random variables on that probability space. In a similar manner, probabilistic Physical Turing Machines and their output distribution can be defined. We leave this as an exercise to the reader.

We will conclude this subsection by a definition of physical probabilistic oracle Turing machines, or PhPOTMs.

Definition 3.16 (Physical Probabilistic Oracle Turing Machines)**.** *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a function. An physical probabilistic oracle Turing machine or PhPOTM M with an oracle for f is a PhPTM with an additional binary tape, the oracle tape, and one additional state, the oracle state. Let x be the content of the oracle tape when M switches ino the oracle state. Then, the following happens within one Turing step: The content of the binary oracle tape is erased and replaced with the value $f(x)$. The function f is also called the oracle function.*

The definition of oracles for PhPTM is basically as the same as for standard TMs. The familiar terms 'oracle call' and 'number of oracle calls' can be used for PhPOTMs in the straightforward way.

3.5 Object Generators and Measuring Devices

We will introduce two special types of PhTMs in this section, which are called object generators (or simply generators) and measuring devices. We will also specify some notation in connection with these devices that is going to make life easier in the upcoming sections.

Intuitively, an object generator should be a PhTM which 'generates' an object: It should take as input a binary string and produce as output one physical object, optionally plus a binary string.

Likewise, the natural purpose of a measuring device is to 'measure' objects: That means it should take as input one physical object, optionally plus a string, and output a binary string plus the unchanged object. We do not allow a measuring device to indicate a possible failure by a special output, though. It is sufficient for our further considerations if the measuring device simply outputs a default value together with the unchanged object in that case.

Definition 3.17 (Object Generators)**.** *Let $\mathcal{U}$ be a probabilistic universe, and $\mathcal{T}$ be a technology in $\mathcal{U}$. A probabilistic PhTM OG in $\mathcal{T}$ is called an object generator in $\mathcal{T}$, if*

$$\mathcal{F}_{OG} : \{0,1\}^* \to \{0,1\}^* \times \mathcal{O}.$$

Definition 3.18 (Measuring Devices)**.** *Let $\mathcal{U}$ be a probabilistic universe, and $\mathcal{T}$ be a technology in $\mathcal{U}$. A deterministic PhTM M in $\mathcal{T}$ is called a measuring device in $\mathcal{T}$, if*

$$\mathcal{F}_M : D_M \to \{0,1\}^* \times \mathcal{O}, \quad \textit{with } D_M \subseteq \{0,1\}^* \times \mathcal{O},$$

and for all $(p, O) \in D_M$:

$$\Pi^2(Out_M(p,O)) = O.$$

Definition 3.19 (Measurement Parameters and Results)**.** *Let M be a measuring device with domain $D_M \subseteq \{0,1\}^* \times \mathcal{O}$. Then, the set $\Pi^1(D_M) \subseteq \{0,1\}^*$ is called the set of measurement parameters or measurement vectors of M, and is often denoted by D^1_M. The elements of $\Pi^1(D_M)$ are called measurement parameters or measurement vectors of M, and are mostly denoted by the terms p or p_i. For any $(p, O) \in D_M$ the value $\Pi^1(Out_M(p,O))$ is called the measurement result of the measurement on O that is executed by M and characterized by p, or simply the measurement result.*

Please note that the expression Π^i denotes the projection of a tuple or a cartesian product set onto its i-th coordinate. Therefore the second condition in definition 3.18 says that the object O is left identical respectively unaltered through the measuring process. This implies that the measurement can be repeated to obtain the same result, which is a key property for later applications.

Obviously this assumption presupposes that only stable classical properties are measured, and that these properties are unchanged through measurement. This is certainly an approximation, but one that seems justified in the realm of our considerations. Further, it excludes quantum measurements. As the systems we consider are non-quantum, this does not harm our framework, though.

Before we proceed to the next section, we will introduce a compact notation for expressions related to measuring devices and measurement results. This notation will make life much easier in the upcoming sections.

Notation 3.20 (Notational Conventions for Measuring Devices)**.** *Let M be a measuring device and $p_1, \dots, p_n$ be measurement parameters of M. For notational ease, we introduce the following abbreviations:*

$$\begin{aligned}
M(p,O) &\overset{\text{def}}{=} \Pi^1(Out_M(p,O)) &&(*)\\
\overline{p} &\overset{\text{def}}{=} (p_1,\dots,p_n)\\
\overline{p}_i &\overset{\text{def}}{=} (p_{(i,1)},\dots,p_{(i,n_i)})\\
M(\overline{p},O) &\overset{\text{def}}{=} (M(p_1,O),\dots,M(p_n,O))\\
M(\overline{p}_i,O) &\overset{\text{def}}{=} (M(p_{(i,1)},O),\dots,M(p_{(i,n_i)},O))\\
\overline{P}(O) &\overset{\text{def}}{=} (\overline{p}, M(\overline{p},O)\\
\overline{P}_i(O) &\overset{\text{def}}{=} (\overline{p}_i, M(\overline{p}_i,O)\\
&= (p_{(i,1)},\dots,p_{(i,n_i)}, M(p_{(i,1)},O),\dots,M(p_{(i,n_i)},O))
\end{aligned}$$

Two comments are in order. First, please recall that in Notation 3.6 we introduced a short form $M(X)$ for the notation $Out_M(x)$, saying that this short form would apply "unless otherwise stated". Equation $(*)$ is the only notable case in which we deviate from the general habit and *do* state otherwise. This is reasonable as we are only interested in the numerical outcome of a measurement, not in the physical part of the output. The introduction of a physical output of the measuring device was mainly a technical condition used to assure that the measured object is not changed through measurement.

Second, please note that the introduced abbreviations in fact "hide" some of the parameters they depend upon. For example, the notational term $\overline{p}$ does not formally exhibit the parameter n any more, in relation to which it is defined. In the same sense, the notational term $\overline{P}(O)$, which depends on the parameter n, the set of measuring vectors $\overline{p}$ and the measuring device M, does not contain any of these terms. This obviously achieves notational compactness, but might appear slightly confusing or even formally incorrect at first glance. We stress, however, that the introduced notation is not to be taken in the sense of a formal definition, but as an abbreviation and simplification. It is to be understood quite mechanical in the sense that any appearance of $\overline{P}(O)$, for example, must in principle be replaced by the longer terms $\overline{p}$, $M(\overline{p}, O)$ or even $(p_1, \ldots, p_n, M(p_1, O), \ldots, M(p_n, O))$.

We will illustrate the intended replacement character of our abbreviations by two examples. If the terms $\overline{P}(O_1)$ and $\overline{P}(O_2)$ are used in one mathematical expression, then they refer to the same set of measuring vectors $\overline{p}$ and the same measuring device M. The terms $\overline{P}_1(O_1)$ and $\overline{P}_2(O_4)$, however, refer to the possibly different sets of measuring vectors $\overline{p}_1 = (p_{(1,1)}, \ldots, p_{(1,n_1)})$ and $\overline{p}_2 = (p_{(2,1)}, \ldots, p_{(2,n_2)})$, the possibly different objects O_1 and O_4, but still the same measuring device M. Similar considerations apply to the use of the abbreviations $\overline{p}$, $\overline{p}_i$ and $\overline{p}_j$.

4 Physical Security of Standard Cryptography

We will now try to formulate the security of standard digital signature schemes on the basis of physical Turing machines. The resulting security notion will be called φ-security. The advantage of φ-security obviously is that it includes physical attacks in its security model. One necessary consequence, as discussed at length in the introuction, is that we have to include the current state of technology in our framework. This makes the φ-security of any given cryptographic signature scheme subject to this state of technology. That fact seems unusual at first glance, but indeed reflects the state of affairs for most cryptographic algorithms: They rely on the intractability of factoring or discrete logarithm problem, and could, in principle, be attacked by quantum algorithms; what saves their practical security is the current state of technology, not a formal mathematical or Turing-machine based argument.

For the convenience of the reader, we start by recollecting some well-known definitions from classical, digital cryptography (see [15]).

Definition 4.1 (Signature Schemes [15]). *A digital signature scheme is a triple (G, Sig, Ver) of probabilistic polynomial-time (Turing) algorithms satisfying the following two conditions:*

1. *On input 1^n, algorithm G (called the key generator) outputs a pair of bit strings.*
2. *For every pair (s,v) in the range of $G(1^n)$, and for every $\alpha \in \{0,1\}^*$, algorithms Sig (signing) and Ver (verification) satisfy*

$$\Pr\left[Ver(v, \alpha, Sig(s, \alpha)) = 1\right] = 1,$$

where the probability is taken over all the internal coin tosses of algorithms Sig and Ver.

Definition 4.2 (Secure Signature Schemes [15]). *A public key signature scheme is secure if for every probabilistic polynomial time oracle machine M, every polynomial p and all sufficiently large n, it holds that*

$$\Pr\left[\begin{array}{l} Ver(v,\alpha,\beta)=1 \textit{ and } \alpha\notin Q_M^{S_s}(v), \\ \quad \textit{where } (s,v)\leftarrow G(1^n) \textit{ and } (\alpha,\beta)\leftarrow M^{S_s}(v) \end{array}\right] < p(n),$$

where the probability is taken over the coin toses of algorithms G, M and Ver as well as over the coin tosses of machine M.

It is well known that secure signature schemes can be constructed under the assumption that factoring large integers is hard. More precisely, the following holds.

Definition 4.3 (Intractability Assumption for Factoring (IAF) [16]). *Let*

$$H_k \overset{\text{def}}{=} \{n = p\cdot q \mid |p| = |q| = k,\ p\equiv 3 \mod 8,\ q\equiv 7 \mod 8\}.$$

Then, the following statement ist called the intractability assumption for factoring: For all probabilistic polynomial Turing machines M there is a negligible function ν such that for all sufficiently large k,

$$\Pr\left[\begin{array}{c|c} M(n) \textit{ outputs a nontrivial} & n \textit{ is sampled uniformly at} \\ \textit{divisor of } n & \textit{random from the set } H_k \end{array}\right] < \nu(k).$$

Now we can state the following theorem by Goldwasser et al. [16].

Theorem 4.4 (Secure Signatures from the IAF [16]). *If the IAF holds, then there is a secure signature scheme.*

Adopting the view discussed in the introduction (section 2.1), we can state that the last theorem makes a conditional statement about the existence of secure signature schemes in the standard security model of complexity based cryptography, whose relevant parts were briefly reviewed preceeding the theorem. This framework, however, does not include computations executed by physical systems. As announced in the introduction, our aim is therefore to transfer Theorem 4.4 into a corresponding statement that involves physical Turing machines. We will not fully complete all steps that are necessary to do so, in particular we skip the corresponding proof of the φ-equivalent of Theorem 4.4. However, we sketch the basic outline by giving the required definitions and theorems, which will provide the reader with a general idea of the direction we are trying pursue.

We start by extending the definition of the security of a (standard) signature scheme through allowing physical oracle Turing machines instead of standard oracle Turing machines in the definition. This approach has the advantage of taking attacks by physical computation machines into account; at the same time it makes necessary to introduce universes and technologies into the picture, relative to which our new notion of security is defined, as discussed in the introduction and section 3.3. The resulting notion of security is called φ-security, *and we will generally use the prefix "φ" in order to denote any concepts or definitions that arise from classical cryptography and security by adding a physical dimension to it.*

Definition 4.5 (φ-Secure Signature Schemes)**.** *Let $\mathcal{U} = (\mathcal{O}, \mathcal{M}, \mathcal{F}, m)$ be a probabilistic universe, and let $\mathcal{T}$ be a technology in that universe. A signature scheme is called φ-secure in $\mathcal{T}$ if for every probabilistic polynomial physical oracle machine M in $\mathcal{T}$, every polynomial p and all sufficiently large n, it holds that*

$$\Pr\left[\begin{array}{c} Ver(v, \alpha, \beta) = 1 \text{ and } \alpha \notin Q_M^{S_s}(v), \\ \text{where } (s, v) \leftarrow G(1^n) \text{ and } (\alpha, \beta) \leftarrow M^{S_s}(v) \end{array}\right] < p(n),$$

where the probability is taken over the random variables G, M and Ver.

If we are to transfer Theorem 4.4 to the realm of φ-security, then we have to reformulate the IAF in the context of PhTMs.

Definition 4.6 (φ-Intractability Assumption for Factoring (φ-IAF))**.** *Let $\mathcal{U} = (\mathcal{O}, \mathcal{M}, \mathcal{F}, m)$ be a probabilistic universe, and let $\mathcal{T}$ be a technology in that universe. Let further*

$$H_k \overset{\text{def}}{=} \{n = p \cdot q \mid |p| = |q| = k,\ p \equiv 3 \mod 8,\ q \equiv 7 \mod 8\}.$$

Then, the following statement ist called the φ-intractability assumption for factoring in $\mathcal{T}$: For all probabilistic polynomial physical Turing machines M in $\mathcal{T}$ there is a negligible function ν such that for all sufficiently large k,

$$\Pr\left[\begin{array}{c|c} M(n) \text{ outputs a nontrivial} & n \text{ is sampled uniformly at} \\ \text{divisor of } n & \text{random from the set } H_k \end{array}\right] < \nu(k)$$

where the probability is taken over the uniform choice of n and the random variable $M(n)$.

Now we can also transfer Theorem 4.4.

Theorem 4.7 (φ-Secure Signatures from the φ-IAF)**.** *Let $\mathcal{U} = (\mathcal{O}, \mathcal{M}, \mathcal{F}, m)$ be a probabilistic universe, and let $\mathcal{T}$ be a technology in that universe. If the φ-IAF holds in $\mathcal{T}$, then then there is a φ-secure signature scheme in $\mathcal{T}$.*

Indeed, the author conjectures that the proof of the theorem in the standard setting carries over basically immediately to the φ-setting, but the proof has not been led yet.

This sequence of definitions and theorems illustrates one aim of PhTMs that we already emphasized in the introduction: To reformulate the security of standard cryptographic schemes with respect to physical computations. Please note that the introdunction of technologies and universes does not much delimit the relevance of our statements: Provided that both the security assumptions and the security statement are formulated according to the same universe and technology, which is very reasonable in practice, the assumptions basically 'cancel out'. We arrive at statements of the following type: Provided that you live in a world with a state of technology that does not allow you to break the φ-IAF, then there are φ-secure signature schemes in your world. This seems a reasonable class of statements, whose generality is certainly not decreased through reference to the notion of a 'technology'. We hope that this can give a first impression of the application of PhTMs to standard cryptography. We will proceed with the application of PhTMs to physical cryptography, where the property of PhTMs to process physical objects becomes important.

5 Unique Objects

5.1 Informal Description of Unique Objects

In the current section we will introduce the notions of unique objects and verifiably unique objects by giving an informal description; formal definitions will follow in the next section.

Intuitively it seems suggestive to call an object unique if it exists only once. Or, more precisely, if the object possesses some unique properties which at a certain timepoint of reference are shared by no other existing object. If unique objects shall be relevant for cryptographic purposes, however, then their uniqueness should not only hold for the moment, but should prevail for some foreseeable time in the future. This should hold even if a unique object and its properties become known to the general public and are subject to active refabrication attempts by fraudsters. That implies that the unique properties of the object should be impossible to *refabricate* by means of current technology, while, at the same time, it should obviously be possible to *produce* unique objects by means of current technology. This seems contradictory.

The contradiction can be resolved by emplyoing random processes in the production of unique objects. These processes each time exert a different influence on the produce, making each object unique. Suitable random processes may, for example, be given by the fabrication variations that inevitably and uncontrollably occur in many nanoscale manufacturing processes.

In the sequel, we will find it convenient to formally distinguish between two different types of unique objects: Standard unique objects and verifiably unique objects.

Standard unique objects (or simply: unique objects) are designed for a situation in which the honest parties have control over the manufacturing process of the object, and compete with an external adversary who attempts to copy it. This situation occurs, for example, if unique objects are used as unforgeable labels for banknotes or other valuable goods. In situations of the like, the focus obviously lies on the uncopyability of the object *after* its production.

Verifiably unique objects, in turn, are utilized in more complex situations, where the honest party does not have control over the manufacturing process of the object, and the manufacturer is regarded as potentially malicious, too. Under such circumstances it does not suffice, roughly speaking, that unique objects are uncopyable for external fraudsters. It becomes also relevant whether the manufacturer himself could have produced more than one specimen of a given object, for example by applying a novel, unexpected trick *during* the production. In situations of the like, the focus lies on the unreproducibility of the object uring *and* after the production process. To the honest user who receives the readily produced objects form the possibly dishonest manufacturer, the desired conclusion for uncopyability is an *ex post* conclusion taking place after the object has been generated.

The feature of verifiability becomes relevant, for example, when we present a novel method to distribute copy protected digital content over online connections.

5.2 Definition of Unique Objects

In a formal definition of the notions of unique objects and verifiably unique objects, the details will be more involved than the introductory characterisations presented in section 5.1. The bottleneck will be the formalisation of vague expressions such as the terms "cannot be reproduced", "properties" or "identical properties". Further, we will have to define our two notions in some probabilistic framework, as the above statements only make sense if small error probabilities are allowed; for example, we have to take the possibility into account that a fraudster produces a copy of some unique object by an extreme amount of sheer luck.

This formalisation will make it necessary to rely on some established asymptotic concepts from complexity theory, such as the distinction between polynomial and super-polynomial time. Therefore, our definitions of unique objects and their variants further have to be asymptotic, too.

This can be achieved by considering so-called unique object systems instead of single unique objects. These are systems which can generate and measure infinitely many "unique objects" in relation to a growing input parameter 1^k.

We start by defining the underlying general notion of an object system.

Definition 5.1 (Object Systems)**.** *Let $\mathcal{U}$ be a universe, and let $\mathcal{T}$ be a technology in $\mathcal{U}$. Let OG be a polynomial object generator in $\mathcal{T}$, and let M be a polynomial measuring device in $\mathcal{T}$. The tuple (OG, M) is called an object system in $\mathcal{T}$ if there is a function $L \in O(n)$ such that*

$$OG : \left\{ 1^k \middle| k \in \mathbb{N} \right\} \longrightarrow \left\{ (\overline{P}(O), O) \;\middle|\; O \in \mathcal{O} \text{ and } \left|\overline{P}(O)\right| \leq L(k) \right\}.$$

We will make a few comments on Definition 5.1. First of all, please note that the definition makes use of the abbreviations introduced in Notation 3.20. In particular, it relies on the replacement character of the abbreviations, as clarified in the discussion following Notation 3.20. As elaborated there, the abbreviation $\overline{P}(O)$ is to be understood as being equal to $\overline{p}, M(\overline{p}, O)$, which clarifies the dependence of any object system on the measuring device M. If further explanation on the used abbrevations is required, the reader is referred again to Notation 3.20.

Then, please note that the definition uses the notion of a measuring device precisely as introduced in Definition 3.18, but employs the notion of object generators in an altered fashion, requiring that they have to meet the described constraints on their domain and range. In particular, we required that the object generators employed in an object system output a set of properties $\overline{P}(O)$ together with the object O, such that $\overline{P}(O)$ is of linear size in the input parameter k. This is demanded with an eye on future applications, where the unique properties of the generated object should be compact and not too long for practicality reasons.

Third, note that any object system can produce infinitely many objects, and is parametrised by a parameter k which is presented to the object generator in unary notation.

The definition of object systems states nothing about any security aspects, however. These will be dealt with in the upconing definitions, where we specify the notion of a unique object system and, eventually, of a verifiably unique object system.

Definition 5.2 (Unique Object Systems)**.** *Let $\mathcal{U}$ be a probabilistic universe, and let $\mathcal{T}$ be a technology in that universe. Let (OG, M) be an object system in $\mathcal{T}$. (OG, M) is called a unique object system in $\mathcal{T}$ if the following* uncloneability condition *holds: For any polynomial φ-TM $CLONE$ in $\mathcal{T}$ with*

$$CLONE : \left\{ (1^k, \overline{P}(O), O) \;\middle|\; O \in \mathcal{O} \text{ and } \overline{p} \in (D^1_M)^* \right\} \longrightarrow \mathcal{O}^2,$$

for any polynomial p and for any sufficiently large k,

$$\Pr\left[\begin{array}{l} \overline{P}(O) = \overline{P}(O_1) = \overline{P}(O_2) \\ \qquad \text{where } (O_1, O_2) \leftarrow CLONE(\overline{P}(O), O) \\ \qquad \text{and } (\overline{P}(O), O) \leftarrow OG(1^k) \end{array}\right] \leq 1/p(k)$$

where the probability is taken over the random outputs of $CLONE$ and OG.

Again, some comments follow. Definition 5.2 is obviously an asymptotic definition in the sense that the PhTMs OG, M and $CLONE$ operate on infinite domains. Like in Definition 5.1, this asymptiticity is achieved by a unary parameter k, which parametrizes the output of OG and is provided to both OG and $CLONE$ as input.

This asymptotic construction is necessary for a number of reasons: First of all and most obviously, because we want to apply the asymptotic concepts of polynomial time and polynomial mass.

There is a more subtle, second reason, too: If OG would operate on a finite domain and hence a finite range, only, then there could be a finite machine $CLONE$ respecting $\mathcal{T}$ which had 'on hold' or 'on store' all finitely many possible outputs of OG. This machine would be suitable as cloning machine: On input $(\overline{P}(O), O)$ it would simply search for an object O' with $\overline{P}(O') = \overline{P}(O)$ among the objects it has 'on hold', and output this object. Provided that the objects are ordered according to their properties, this search could presumably be carried out quickly. Hence, we need to restrict the size of the physical systems that are employed by PhTM $CLONE$, so that they can only have some small fraction of all possible outcomes of OG 'on hold'. A proper choice for that restriction of size is the notion of polynomial mass. Again, this presumes an asymptotic treatement, and also implicitely assumes that $OG(1^k)$ can produce super-polynomially (in k) many different objects.

Finally, please note that Definition 5.2 perhaps surprisingly says nothing about the 'uniqueness' of the generated objects, but only states some 'uncloneability condition'. The next propositition shows, however, that some sort of uniqueness property is already implicit in the uncloneability condition.

Proposition 5.3. *Let $\mathcal{U}$ be a universe, $\mathcal{T}$ be a technology, and $OS = (OG, M)$ be a unique object system in $\mathcal{T}$. Let further p and q be polynomials, and let $OG_1, \ldots, OG_{p(k)}$ be object generators in $\mathcal{T}$. Then, the following holds:*

1. *For all polynomials r and for all sufficiently large k,*

$$\Pr\left[\begin{array}{l} \exists\, i \neq j: \ \overline{P}_i(O_i) = \overline{P}_j(O_j) \\ \qquad \textit{where } \big(\overline{P}_i(O_i), O_i\big) \leftarrow OG(1^k) \textit{ for } i = 1, \ldots, q(k) \end{array}\right] \leq 1/r(k).$$

2. *For all polynomials r and for all sufficiently large k,*

$$\Pr\left[\begin{array}{l} \exists\, (i,m) \neq (j,l): \ \overline{P}_{(i,m)}(O_{(i,m)}) = \overline{P}_{(j,l)}(O_{(j,l)}) \\ \qquad \textit{where } \big(\overline{P}_{(j,l)}(O_{(j,l)}), O_{(j,l)}\big) \leftarrow OG_j(1^k) \\ \qquad \textit{for } j = 1, \ldots, p(k) \textit{ and } l = 1, \ldots, q(k). \end{array}\right] \leq 1/r(k).$$

Proof. We will only show the first statement; the proof of the second statement is similar. For the sake of contradiction, we make the following assumption:

Contradiction Assumption: OS is a unique object system, but for some polynomial r and for infinitely many k it holds that

$$\Pr\left[\begin{array}{l} \exists\, i \neq j: \ \overline{P}_j(O_j) = \overline{P}_i(O_i) \\ \qquad \text{where } \big(\overline{P}_i(O_i), O_i\big) \leftarrow OG(1^k) \text{ for } i = 1, \ldots, q(k) \end{array}\right] > 1/r(k).$$

Then, we construct a PhPTM $CLONE$ as follows:

```
Machine CLONE:
  Input  $(1^k, \overline{P}(O), O)$
    Set  $(\overline{P}_i(O_i), O_i) \leftarrow OG(1^k)$   for $i = 1, \ldots, q(k)$
    If  $\nexists i: \overline{P}_i(O_i) = \overline{P}(O)$
      then  output “failure” and abort.
  Output  $(\overline{P}(O_1), O_1, \overline{P}(O_2), O_2)$
```

By the construction of $CLONE$ and statement 1 it is clear that for any polynomial p and for any sufficiently large k,

$$\Pr\left[\begin{array}{l} \overline{P}(O) = \overline{P}(O_1) = \overline{P}(O_2) \\ \qquad \text{where } (O_1, O_2) \leftarrow CLONE(\overline{P}(O), O) \\ \qquad \text{and } (\overline{P}(O), O) \leftarrow OG(1^k) \end{array}\right] > 1/r(k)$$

where the probability is taken over the random outputs of $CLONE$ and OG.

This contradicts the assumption that (OG, M) was a unique object system, and completes the proof.

□

We will now turn to the definition of verifiably unique object systems, and start by a thorough demarcation to the notion of unique object systems.

The definition of unique object systems asserts that there is no external machine which clones the objects after they have been produced, and it also asserts by Proposition 5.3 that the very object generator OG of the unique object system (OG, M) produces two identical objects with negligibly small probability only. Further, Proposition 5.3 asserts that any other object generator, too, has got a very small probability to reproduce by chance an object that has earlier been produced by OG. Definition 5.2 does not guarantee, however, that there might not be *another* object generator OG' different from OG which can produce objects of similar type as OG, but which is capable of generating more than one specimen of each produced object! Any such specimen produced by OG' would by itself look like a unique object to an external observer, while, in fact, the manufacturer holds possession of a "twin" with identical properties.

Let us further clarify the difference between our hypothetical machine OG and the attacker $CLONE$ as specified in the uncloneability condition of Definition 5.2: The machine $CLONE$ arbitrary object produced by someone else, namely the object generator OG, and attempts to copy it. Our hypothetical machine OG', however, generates the objects by itself, and tries to design the production process in such a way that two or more identical objects result, each of which on its own looks like a unique object. Hence, Definition 5.2 forbids the existence of a succesful attacker $CLONE$, but does not rule out the existence of the described object generator OG'.

It is important to state that this problem is far from trivial or merely theoretical. It appears in any application where the user of some unique object does not trust the manufacturer of the object, and wants to be sure that no other person, *including* the manufacturer, can possess or produce an identical object. The definition of unique object systems cannot be used to account for these cases, as it merely provides security guarantees against external fraudsters. We need a new notion, which we call verifiably unique object systems, and a new definition.

If that new definition is to make sense, it has to include a mechanism by which the honest user can actually check that a given, supposedly unique object really is unique. We hence have to select a form that this check is supposed to take. There are two suggestive choices: The honest user could try to check the uniqueness by some physical actions, like inspection of the object. Or, alternatively, by merely elaborating on the numerical properties of the object, without physical inspection. The latter choice is more convenient, as it allows the practically advantageous possibility that the numerical properties of the object are measured by another party instead of the honest user.

The following definition formally expresses the properties of verifiably unique object systems as discussed.

Definition 5.4 (Verifiably Unique Object Systems)**.** *Let $\mathcal{U}$ be a probabilistic universe, and let $\mathcal{T}$ be a technology in that universe. Let (OG, M) be an object system in $\mathcal{T}$. (OG, M) is called a verifiably unique object system in $\mathcal{T}$ if the following two conditions* *are met:*

1. Manufacturer Resistancy Condition: *For any polynomial PhTM $TWIN$ in $\mathcal{T}$ with*

$$TWIN : \{ 1^k \,|\, k \in \mathbb{N} \} \longrightarrow \{ \big(\overline{P}(O_1), O_1, \overline{P}(O_2), O_2\big) \,|\, O_1, O_2 \in \mathcal{O},\ \overline{p} \in (D_M^P)^* \},$$

for any polynomial p and for all sufficiently large k, it holds that

$$\Pr \left[\begin{array}{c} \overline{P}(O_1) = \overline{P}(O_2) \text{ and} \\ \big(\overline{P}(O_1), O_1\big), \big(\overline{P}(O_2), O_2\big) \in \mathsf{Range}(OG) \setminus \{(\lambda, \lambda_O)\} \\ \text{where } \big(\overline{P}(O_1), O_1, \overline{P}(O_2), O_2\big) \leftarrow TWIN(1^k) \end{array} \right] \leq 1/p(k)$$

where the probability is taken over the random output of $TWIN$.

2. Verifiability Condition: *There is a polynomial time probabilistic Turing machine OV, called the object verifier, which decides membership in*

$$\Pi^1(\mathsf{Range}(OG)) \setminus \{(\lambda, \lambda_O)\}.$$

That is, given a string $x \in \{0,1\}^*$, OV decides in probabilistic polynomial time whether there is an object $O \in \mathcal{O}$ such that

$$(x, O) \in \mathsf{Range}(OG) \setminus \{(\lambda, \lambda_O)\}.$$

Once more, some comments follow. We will start with some formalities: First, note again that again that the abbreviations introduced in Notation 3.20 are used extensively in the definition. If anything is doubtful, we refer again to Notation 3.20.

Then, it is obvious that the definition just like the previous definitions is asymptotic, and that the behavior of OG is asymptotically parametrised throughthe security parameter k. This parameter is also presented to the adversarial machine $TWIN$.

Coming to more substantial issues, we would like to re-emphasize the two different parts of Definition 5.4, which relate to the two basic properties of verifiably unique object systems. Part one expresses the manufacturer resistance condition, which states the requirement that there is no alternative production process that generates the same objects as OG, but which is capable of manufacturing two or more identical specimen of each produced object. In part two, the verifiability condition asserts that the manifacturer resistance can be efficiently verified by a user by merely juding the numerical properties of the produced objects.

It is interesting to state in this context that the formal expression of the property of manufacturer resistance as in Definition 5.4 answers an issue left open in [12, 11], where the property of manufacturer resistance is introduced informally and used subsequently without a formal definition.

We would like to remark in this context, however, that we feel that the practical relevance of the notion of manufacturer resistance is significantly decreased if it used without a corresponding verifiability condition as in Definition 5.4. Any user distrusting the manufacturer must be able to verify *by himself* that a given object was produced in a manufacturer resistant way, as he apparently will not be willing to trust the manufacturer's assertion that this was the case. This, then, requires some verification mechanism. If, on the other hand, the manufacturer *is trusted* by the user, no manufacturer resistance property is needed in the first place. This problem of verifying manufacturer resistance in modern PUF-protocols has just recently been picked up again by Rührmair and van Dijk at Oakland 2013 in introducing the so-called "bad PUF model" [27].

We suggest therefore contrary to [12], that the notion of manufacturer resistance should in general only be used in connection with a corresponding verifiability condition, and did not define manufacturer resistance or manufacturer resistant object systems, as independent notions of their own.

There is another issue about the manufacturer resistance condition that is worth discussing: Why do we exclude the output (λ, λ_O) from the range of OG?

The reason lies in the probabilistic nature of the production process executed by OG. Recall that the aim of OG is to produce an object which is unique and cannot be reproduced. By sheer bad luck, however, it could be the case that the manufactured object happens to be trivial and easy to reproduce. In that case, a natural response of OG would simply be to repeat the production process. However, there again is a small probability that the result is unsatisfactory, and with *very* small probability, OG could even be unable to generate a satisfactory object within the prespecified polynomial time bound.

In that case, OG simply generates the failure output (λ, λ_O), that is, it halts with its binary and its physical tape being empty. This output then certainly is trivial to reproduce and has to be excluded from the allowed outputs of $TWIN$ in order not to spoil the definition. If it was not excluded, then $TWIN$ could on any input 1^k produce the output $(\lambda, \lambda_O, \lambda, \lambda_O)$, thereby breaking any verifiably unique object system.

Likewise, if we do not allow OG to produce a failure output that is excluded from the allowed range of $TWIN$, then there might be easy-to-reproduce objects in the range of OG, and $TWIN$ could confine itself to generating two copies of one of these easy-to-reproduce objects. Thereby it could break the supposedly verifiably unique object system (OG, M).

The notion of being a verifiably unique object system seems stronger than being a unique object system. Intuitively one would assume that the former implied the latter in some sense; this is stated more precisely and proved in the upcoming proposition.

Proposition 5.5 (Verifiably Unique Object Systems are Unique Object Systems). *Let $\mathcal{U}$ be a universe, and let $\mathcal{T}$ be a technology. Let (OG, M) be a verifiably unique object system in $\mathcal{T}$. Then (OG, M) is also a unique object system in $\mathcal{T}$.*

Proof (Sketch). We argue by contradiction, assuming the following:

Contradiction Assumption: (OG, M) is a verifiably unique object system in $\mathcal{T}$, but not a unique object system in $\mathcal{T}$.

By the definition of a unique object system, the latter part of the contradiction assumption implies the following:

Statement 1: There is a polynomial PhPTM *CLONE* such that for infinitely many k and a fixed polynomial p^* it holds that

$$\Pr\left[\begin{array}{l}\overline{P}(O) = \overline{P}(O_1) = \overline{P}(O_2) \\ \qquad \text{where } (O_1, O_2) \leftarrow \mathit{CLONE}(\overline{P}(O), O) \\ \qquad \text{and } (\overline{P}(O), O) \leftarrow \mathit{OG}(1^k)\end{array}\right] > 1/p^*(k),$$

where the probability is taken over the random outputs of *CLONE* and *OG*.

The machine *CLONE* can then be used in order to build a PhPTM *TWIN* which violates the property that (OG, M) is a unique object system. We define *TWIN* as follows:

Machine *TWIN*:

```
Input  $1^k$
    Set  $\big(\overline{P}(O), O\big) \leftarrow OG(1^k)$
    Set  $(O_1, O_2) \leftarrow CLONE\big(1^k, \overline{P}(O), O\big)$
Output  $\big(\overline{P}(O_1), O_1, \overline{P}(O_2), O_2\big)$
```

It is obvious from the definition of *TWIN* that if *CLONE*, *OG* and M are polynomial PhPTM in $\mathcal{T}$, then so is *TWIN*. Further, one can see that due to the properties of *CLONE* and *OG* one can derive from statement 1 thatit holds for infinitely many k and the polynomial P^* from statement 1 that

$$\Pr\left[\begin{array}{l}\overline{P}(O_1) = \overline{P}(O_2) \\ \qquad \text{where } \big(\overline{P}(O_1), O_1, \overline{P}(O_2), O_2\big) \leftarrow \mathit{TWIN}(1^k)\end{array}\right] > 1/p^*(k).$$

This implies that (OG, M) violates the manufacturer resistance condition and hence is no verifiably unique object system. This is at odds with the contradiction hypothesis and completes the proof. □

An obvious question to ask is whether the converse of Proposition 5.5 also holds. This question is not straightforward to answer.

Intuitively, one would (probably) regard verifiably unique object systems a stronger notion than unique object systems, suspecting that there are unique object systems which are not verifiably unique object systems. However, it seems very hard with current techniques to prove this assumption uncoditionally, because this would imply an unconditional proof that some system is a unique object system in the first place. Such proofs – like proofs that a given function is a one-way function – seem to be beyond the current knowledge in theoretical computer science, if led unconditionally and without restricting the universes or technologies artificially.

Still, one can (without proof) imagine some unique object systems which are likely not to meet the manufacturer resistance condition of verifiably unique object systems. This backs the assumption that the two notions are different. As an example, take as OG a machine which produces on input 1^k a unique, random mask. Further, imagine that OG subsequently uses that mask in order to manufacture *one* physical object in such a way that its properties are a deterministic and repeatable outcome of the mask structure. Examples might include the printing masks used for banknotes, which are unique in their microstructure, or lithographic masks in semiconductor technology. To complete the object system, take as measuring device M some PhTM which measures the properties of the manufactured systems.

Then, it is obvious under these premises that we could just as well build an object generator OG' which works as OG, but uses the mask to produce two or even more objects with identical properties. Hence, (OG, M) does not meet the manufacturer resistance condition and is no verifiably unique object system.

Further, note that given one object produced by either OG or OG', it is impossible to tell by whether it was produced by OG or OG', whence necessarily also the verifiability condition is violated.

Another, quite convincing example arises in the context of chemistry. Consider a very complex chemical or biochemical solution, for example a complex, randomly produced DNA-solution. Such random, highly entropic solutions have been suggested recently in the context of DNA-based cryptography, for example in Nature magazine and elsewhere [9, 13, 8]. By stirring the solution carefully in order to achieve an isotropic distribution of the constituents, and by subsequently extracting equal parts of the volume, one gets "little identical copies" of the original solution. These "little copies" share the same relative distribution of constituents as the original solution. Hence, on seeing a randomly looking DNA-solution, there is no way to tell whether this volume was once part of a larger solution with an identical constituent distribution, or whether this volume is already the whole original and randomly produced solution.

Therefore an object system producing such random DNA-solution is a convincing candidate for a unique object system which is no verifiably unique object system. Its practical implementation seems quite realistic, as the relative concentrations of the constituents can be tested reliably by established biochemical techniques, for example by use of DNA-chips.

The example of complex DNA-solutions can hence be used to maintain that the notions of uniqueness and verifiably uniqueness as defined earlier are different, and that using two different definitions is well justified also from a practical point of view.

6 Labeling Schemes

6.1 Definitions

In the sequel, we will deal with one of the main applications of unique object systems, which is the unforgeable labeling of valuable goods. We imagine that this task is executed by two machines with the following subtasks:

Machine 1, which is capable of producing physical tokens, the *labels.*

Machine 2, which is capable of deciding whether a given physical token is a valid label produced by Machine 1, or not.

Further, we can imagine that a third machine plays a role, which produces a secret key by which the other two machines can be configured or personalised.

Machine 3, which can produce some binary information by which Machine 1 and Machine 2 can be configured or personalised, respectively, for different users.

This leads to the following definition:

Definition 6.1 (Labeling Schemes)**.** *Let $\mathcal{U}$ be a probabilistic universe and $\mathcal{T}$ a technology in $\mathcal{U}$. A labeling scheme in $\mathcal{T}$ is a triple (C, LG, T) of probabilistic polynomial mass PhTMs in $\mathcal{T}$ with the following properties:*

1. *C, called the configurator, runs in polynomial time. It produces on input 1^k a pair of binary strings (p, t) as output. Thereby p is the configurating information that is later provided to the producer, and t is the configurating information provided to the tester.*
2. *LG, called the label generator, runs in polynomial time. It produces on input $(1^k, p)$ an output $(x, O) \in \{0,1\}^* \times \mathcal{O}$. The output (x, O) is called a label, and often denoted by the letter L.*
3. *T is called the tester and runs in polynomial time. It takes as input a triple (t, x, O) from the set $\{0,1\}^* \times \mathcal{O}$ and produces a binary output with the property that for every k and for every pair (p, t) in the range of $C(1^k)$,*

   $$\Pr\left[T(t, LG(1^k, p)) = 1\right] = 1,$$

   where the probability is taken over the random output of LG and T.

Note that this definition for the first time uses the capability of PhTMs to process physical objects as input and output. It says nothing about the security properties of labeling schemes, though; these are dealt with in the next definition. The security model formulated there is reminiscent of the security of digital signatures under known message attack: We assume that the attacker gets a polynomial number of t valid labels $L_1, \ldots, L_t$ as input, and is asked to produce $t+1$ valid labels $L'_1, \ldots, L'_{t+1}$ as output. Put differently, he has to produce one more label than he was presented with, and the new labels can differ from the old ones, as long as they are recognized by the tester as valid. The following definition says this more formally.

Definition 6.2 (Secure Labeling Schemes)**.** *Let $\mathcal{U}$ be a universe and $\mathcal{T}$ be a technology in $\mathcal{U}$, and let (C, LG, T) be a labeling scheme in $\mathcal{T}$. (C, LG, T) is called a secure labeling scheme in $\mathcal{T}$ if for any probabilistic polynomial PhTM FAKE in $\mathcal{T}$ and for any polynomial p there is a negligible function ν such that*

$$\Pr\left[\begin{array}{l} T(t, L'_i) = 1 \quad \text{for } i = 1, \ldots, p(k)+1 \\ \qquad \text{where } (L'_1, \ldots, L'_{p(k)+1}) \leftarrow \mathit{FAKE}(L_1, \ldots, L_{p(k)}), \\ \qquad L_1 \leftarrow LG(1^k, p),\ \ldots,\ L_{p(k)} \leftarrow LG(1^k, p) \text{ and } (p, t) \leftarrow C(1^k) \end{array}\right] < \nu(k).$$

The probability is taken over the random outputs of C, LG, T and FAKE.

After we have presented the necessary definitions of labeling schemes and secure labeling schemes, we will show how a labeling scheme can be based on unique object systems and digital signatures.

6.2 Standard Labeling by Unique Objects and Digital Signatures

It seems suggestive to use unique objects as unforgeable labels, as they fulfill the uncloneability condition of Definition 5.2. Nevertheless, there is a problem with this approach: All objects produced by a UOS are different and 'random' in some way; how shall the tester make a difference between a random object produced by a fraudster and a random object generated by the legitimate manufacturer?

This question can be resolved by combining a standard approach from mathematical cryptography with our new notion of unique objects; such hybrid techniques are rather typical for φ-cryptography. The idea is as follows: The legitimate manufacturer is given a secure signature scheme (G, Sig, Ver) and a unique object system (OG, M). He uses OG to produce a unique object together with some measuring vectors $p_1, \ldots, p_n$, and obtains some corresponding measuring results $M(O, p_1), \ldots, M(O, p_n)$. Then, he uses his secret signature key to sign the properties of the unique object, which proves that this very object originated from him, not from the fraudster. The details are as follows.

Construction 6.3 *Labeling Schemes from Signature Schemes and Object Systems*

Let $DSS = (G, Sig, Ver)$ be a signature scheme, and $OS = (OG, M)$ be an object system. We construct a labeling system $LS = (C, LG, T)$ from DSS and OS as follows:

Configurator C: We take as configurator C the key generator G of the signature scheme. Hence, $C(1^k) = (p, t) \overset{\text{def}}{=} G(1^k) = (s, v)$ for all $k \in \mathbb{N}$.

Label Generator P: The Label Generator LG is constructed from the object generator OG of the object system and the measuring apparatus M of the object system in the following way: LG takes as input p (the partial output of the configurator). Then, it runs $OG(1^k)$. This produces an output $(\overline{P}(O), O)$. Then, the Label Generator produces a digital signature string $S = Sig(s, \overline{P}(O))$. It outputs the label $L = (x, O) \overset{\text{def}}{=} (\overline{P}(O), S, O)$.

TESTER T: The tester is constructed from the measuring device M of the object system and the verifier V from the signature scheme. Given a label L of the form $L = (\overline{P}(O)', S', O')$, where $\overline{P}(O)'$ is of the form $\overline{P}(O)' = \overline{p}\,', M(\overline{p}, O)'$, it proceeds as follows:

(a) It checks whether $Ver(t, \overline{P}(O)', S') = 1$.
(b) It examines by use of M whether $M(\overline{p}, O)' = M(\overline{p}\,', O')$.

If conditions (a) and (b) are met, then the tester regards the label as valid and outputs "1", otherwise it outputs "0".

As the above construction will be used often, we give an own name to it.

Definition 6.4. *Let OS be an object system, and DSS be a signature scheme. Then we denote the labeling scheme LS constructed from OS and DSS as described in the previous construction 6.3 as $SL(OS, DSS)$, and call it the standard labeling scheme (constructed) from OS and DSS.*

Before we turn to the proof of security of the standard labeling scheme, we will describe how the scheme is used in practice. This protocol will say nothing new compared to construction 6.3, but it will bring some practical aspects of the standard labeling scheme illustratively to the point.

Protocol 6.5: OFFLINE LABELING VIA UNIQUE OBJECTS

PREREQUISITES AND SITUATION:

1. There is an institution CA (the 'Central Authority') which issues the labels.

2. The CA holds a presumed unique object system (OG, M), and a presumed secure signature scheme (G, Sig, Ver).

3. The CA is capable of storing digital information on a physical object, for example by using a barcode-like encoding technique or by an RFID-chip.

4. There are a number of testing devices $T_1, \ldots, T_n$, whose purpose is to decide whether a given label is genuine. Each of these devices contains a measuring device M_i, which is "equivalent" to the measuring apparatus M: It can execute the same measurements and will obtain the same results as M.

5. Each of the testing devices is equipped with machinery to read out information that was stored on a physical object by the CA.

6. The CA has chosen security parameters l and m by which it runs the unique object system and the signature scheme.

7. The CA has run the key generator G of the signature algorithm in order to produce a key pair $(s, v) \leftarrow G(1^l)$. The key s is secret and only known to the CA; the key v is public and has been distributed to all testing devices.

Scheme for Labeling a Product P:

1. The CA runs the machine OG of the object system with input 1^m to produce an output $OG(1^m) = (\overline{p}, M(\overline{p}, O), O)$.
2. The CA signs the information $\overline{p}, M(\overline{p}, O), PD$ by the signing algorithm of the signature scheme and the signing key s. Here, PD denotes some further product data useful for handling or shipping the product. Thereby it creates a string
$$S \overset{\text{def}}{=} Sig(s, \overline{p}, M(\overline{p}, O), PD).$$
3. The CA sticks the object O to the product. Further, it stores the information $\overline{p}, M(\overline{p}, O), PD, S$ on the product.

Scheme for Testing a Label:

1. Some tester T_i is presented with a product that contains a label consisting of the following items:
   (a) A physical object O'.
   (b) Some numerical data of the form $\overline{p}\,', M(\overline{p}, O)', PD', S'$.

   If the testing device finds that the label does not have this form, then it concludes that the label is faked and aborts.
2. The tester checks whether the signature S' is valid by testing if
$$Ver(v, (\overline{p}\,', M(\overline{p}, O)', PD'), S') = 1.$$
3. The tester uses the measuring device M to check if
$$M(\overline{p}\,', O') = M(\overline{p}, O)'$$
4. The tester regards the label as valid if and only if the checks described in the last two steps were positive.

This concludes our description of the standard labeling scheme. In the next section we will present a formal proof of the security of the standard labeling scheme. That proof will be led in the formal framework that we have developed over the last sections.

6.3 Security Proof for the Standard Labeling Scheme

We start by a notational convention.

Notation 6.6. *Let* $L = \big(\overline{p}, M(\overline{p}, O), Sig(s, \overline{p}, M(\overline{p}, O)), O\big)$ *be a label generated by some standard labeling scheme* $LS = SL(OS, DSS)$*. Then we introduce the following notations:*

$$\begin{aligned}
\mathsf{Obj}(L) &\overset{\text{def}}{=} O \\
\mathsf{Par}(L) &\overset{\text{def}}{=} \overline{p} \\
\mathsf{Pro}(L) &\overset{\text{def}}{=} (\overline{p}, M(\overline{p}, O)) \\
&= \big(\mathsf{Par}(L), M(\mathsf{Par}(L), O)\big) \qquad (*) \\
\mathsf{Sig}(L) &\overset{\text{def}}{=} Sig\big(s, \overline{p}, M(\overline{p}, O)\big)
\end{aligned}$$

The following, basically simple observation will be one of the keys to the proof.

Lemma 6.7 (One Faked Label means One New Signature or One Cloned Object)**.** *Let* $LS = SL(OS, DSS) = (C, LG, T)$ *be some standard labeling scheme, and let* $L_1, \ldots, L_m$ *and* $L'_1, \ldots, L'_n$ *with* $m < n$ *be "valid labels" of* LS*. That is, for some* $(s, v) \in \mathsf{Range}\,(G)$*, where* G *is the generator of* DSS*,*

$$\begin{aligned}
T(v, L_i) &= 1 \quad \text{for } i = 1, \ldots, m, \quad \text{and} \\
T(v, L'_i) &= 1 \quad \text{for } i = 1, \ldots, n.
\end{aligned}$$

Then, at least one of the following two statements holds:

(1) The labels $L'_1, \ldots, L'_n$ *contain two identical "copies" of one certain object contained in the labels* $L_1, \ldots, L_m$*.*

Or, more formally: $\exists\, i, j \in \{1, \ldots, n\}$, $k \in \{1, \ldots, m\}$:
$M\,(\mathsf{Par}(L_k), \mathsf{Obj}(L'_i)) \;=\; M\,(\mathsf{Par}(L_k), \mathsf{Obj}(L'_j)) \;=\; M\,(\mathsf{Par}(L_k), \mathsf{Obj}(L_k))$.

(2) One of the labels $L'_1, \ldots, L'_n$ *contains a "new" valid digital signature that is not contained in any of the labels* $L_1, \ldots, L_m$*.*

Or, more formally: $\exists\, i \in \{1, \ldots, n\} :\ Ver\,(v, \mathsf{Pro}(L'_i), \mathsf{Sig}(L'_i)) = 1$
and $\nexists\, j \in 1, \ldots, k :\ \mathsf{Pro}(L'_i) = \mathsf{Pro}(L_j)$

Proof. We lead the proof by considering two sets PRO and PRO'. These sets are defined as follows:

$$PRO \overset{\text{def}}{=} \{\mathsf{Pro}(L_i) \mid i = 1, \ldots, m\},$$

and

$$PRO' \overset{\text{def}}{=} \{\mathsf{Pro}(L'_i) \mid i = 1, \ldots, n\}.$$

In other words: PRO is the set of the properties of the objects contained in the labels $L_1, \ldots, L_m$, and likewise for PRO' with the labels $L'_1, \ldots, L'_n$.

We distinguish between two cases:

Case 1: *$PRO \supseteq PRO'$.* As $m < n$, this implies that there must be indices $i, j \in \{1, \ldots, n\}$ such that $\mathsf{Pro}(L'_i) = \mathsf{Pro}(L'_j)$, and that there must be a further index $k \in \{1, \ldots, m\}$ such that $\mathsf{Pro}(L_k) = \mathsf{Pro}(L'_i) = \mathsf{Pro}(L'_j)$. This implies by the definition of Pro (equation $(*)$ in Notation 6.6) that

$$\mathsf{Par}(L_k) = \mathsf{Par}(L'_i) = \mathsf{Par}(L'_j),$$

and that

$$M(\mathsf{Par}(L_k), \mathsf{Obj}(L_k)) = M(\mathsf{Par}(L'_i), \mathsf{Obj}(L'_i)) = M(\mathsf{Par}(L'_j), \mathsf{Obj}(L'_j)).$$

Together, this implies that

$$M(\mathsf{Par}(L_k), \mathsf{Obj}(L'_i)) \;=\; M(\mathsf{Par}(L_k), \mathsf{Obj}(L'_j)) \;=\; M(\mathsf{Par}(L_k), \mathsf{Obj}(L_k)).$$

Hence, statement (1) of the lemma holds.

Case 2: *$PRO \subsetneq PRO'$.* Then there is a label L'_i in PRO' such that $\mathsf{Pro}(L'_i) \neq \mathsf{Pro}(L_j)$ for all $j = 1, \ldots, m$. As $T(v, L'_i) = 1$, it holds due to the construction of labels and the tester T of the standard labeling scheme that $Ver(v, \mathsf{Pro}(L'_i), \mathsf{Sig}(L'_i)) = 1$. Therefore, statement (2) of the lemma is fulfilled.

As the cases cover all possibilities, this shows that under the given premises at least one of the statements (1) and (2) holds. This completes the proof. □

Before we tackle the Main Theorem, we will prove another lemma. It states that a generalization of the notion of a unique objects system is equivalent to the original definition. We start by defining the generalization.

Definition 6.8 (p-Unique Object Systems). *Let $\mathcal{U}$ be a universe, and let $\mathcal{T}$ be a technology. Let $OS = (OG, M)$ be an object system, and p be a polynomial. OS is called a p-unique object system in $\mathcal{T}$, if the following holds: For any polynomial φ-TM p-CLONE in $\mathcal{T}$, for all polynomials q and for all sufficiently large k,*

$$\Pr\left[\begin{array}{l} \exists\, i \neq j,\, l:\; \overline{P}_l(O'_i) = \overline{P}_l(O'_j) = \overline{P}_l(O_l) \\ \textit{and } \forall\, l\; \exists\, i:\; \overline{P}_l(O_l) = \overline{P}_l(O'_i) \\ \qquad \textit{where } (O'_1, \ldots, O'_{p(k)+1}) \\ \qquad \leftarrow \textit{p-CLONE}\left(1^k, \overline{P}_1(O_1), O_1, \ldots, \overline{P}_{p(k)}(O_{p(k)}), O_{p(k)}\right) \\ \qquad \textit{and } (\overline{P}_i(O_i), O_i) \leftarrow OG(1^k) \textit{ for } i = 1, \ldots, p(k) \end{array}\right] \leq 1/q(k)$$

The probability is taken over the random outputs of p-CLONE and OG.

It can be seen easily that the notion of a $\mathsf{p\text{-}UOS}$ includes the notion of a UOS; indeed, a UOS is nothing more than a $\mathsf{2\text{-}UOS}$, where the polynomial p is equal to the constant 2. The other direction will be shown in the next lemma, proving that the two notions coincide.

Lemma 6.9 (Equivalence of UOS and p-UOS). *Let $\mathcal{U}$ be a universe and $\mathcal{T}$ be a technology. Let further OS be an object system in $\mathcal{T}$. Then, the following statements are equivalent:*

1. *OS is a unique object system in $\mathcal{T}$.*
2. *OS is a p-unique object system in $\mathcal{T}$ for all polynomials p with $p(n) \geq 2 \; \forall n \in \mathbb{N}$.*

Proof. The implication "$\Leftarrow$" is clear from the two respective definitions (Def. 5.2 and 6.8): Simply take $p(n) = const. = 2$ for all $n \in \mathbb{N}$.

The other direction "$\Rightarrow$" needs more elaboration. Let OS be an object system in $\mathcal{T}$. We lead the proof by contradiction, making the following assumption:

Contradiction Assumption: OS is a unique object system in $\mathcal{T}$, but for some polynomial p with $\big(p(n) \geq 2$ for all $n \in \mathbb{N}\big)$, OS is not a p-unique object system in $\mathcal{T}$.

The contradiction assumption implies the following:

Statement (1): There is a polynomial PhTM $\boldsymbol{p}$-*CLONE* such that for the polynomial p from the contradiction assumption, some polynomial q and infinitely many k,

$$\Pr\left[\begin{array}{l} \exists\, i \neq j,\, l: \; \overline{P}_l(O'_i) = \overline{P}_l(O'_j) = \overline{P}_l(O_l) \\ \text{and } \forall\, l \;\exists\, i: \; \overline{P}_l(O_l) = \overline{P}_l(O'_i) \\ \qquad \text{where } (O'_1, \ldots, O'_{p(k)+1}) \\ \qquad \leftarrow \boldsymbol{p}\text{-}CLONE\big(1^k, \overline{P}_1(O_1), O_1, \ldots, \overline{P}_{p(k)}(O_{p(k)}), O_{p(k)}\big) \\ \qquad \text{and } \big(\overline{P}_i(O_i), O_i\big) \leftarrow OG(1^k) \;\; \text{for } i = 1, \ldots, p(k) \end{array}\right] > 1/q(k),$$

where the probability is as taken over the random outputs of $\boldsymbol{p}$-*CLONE* and *OG*.

The physical Turing machine $\boldsymbol{p}$-*CLONE* addressed in statement (1) "clones" for infinitely many k with non-negligible probability one of the $p(k)$ input objects. If we could use $\boldsymbol{p}$-*CLONE* to set up a second physical TM *CLONE* that does the same for *one* input object, then OS could be no unique object system. This would provide the sought contradiction and complete the proof.

The rest of the proof basically consists in unfolding this simple thought, which will require some technical elaboration. First of all, we need to find an appropriate way to construct the algorithm *CLONE*; then, we need to prove that the success probability of that algorithm is non-negligible.

We start by defining *CLONE*. It depends on the polynomial p whose existence is guaranteed by statement (1), and the PhTM $\boldsymbol{p}$-*CLONE*.

```
MACHINE $CLONE$:
  Input $(1^k, \overline{P}(O), O)$
      Choose $i_0$ uniformly at random from $\{1, \ldots, p(k)\}$
      Set $(\overline{P}(O_{i_0}), O_{i_0}) \leftarrow (\overline{P}(O), O)$
      Set $(\overline{P}(O_i), O_i) \leftarrow OG(1^k)$    for $i = 1, \ldots, i_0 - 1, i_0 + 1, \ldots, p(k)$
      Set $(\overline{P}(O'_1), O'_1, \ldots, \overline{P}(O'_{p(n)+1}), O'_{p(n)+1})$
          $\leftarrow p\text{-}CLONE(\overline{P}(O_1), O_1, \ldots, \overline{P}(O_{p(n)}), O_{p(n)})$
      If $\left(\begin{array}{l} \exists\, i \neq j \in \{1, \ldots, p(n)+1\},\ k \in \{1, \ldots, p(n)\} : \\ i_0 = k \text{ and } \overline{P}_k(O'_i) = \overline{P}_k(O'_j) = \overline{P}_k(O_k) \end{array}\right)$
          then proceed, else output “failure” and abort.
  Output $\overline{P}(O_{i_0}), O'_i, O'_j$
```

We would now like to calculate the probability that $CLONE$ is “successful” for a certain input $(1^k, \overline{P}(O), O)$. Obviously we cannot determine that probability in an absolute sense, but only in relation to the “success probability” of the algorithm $p\text{-}CLONE$. To that aim, we introduce a random variable CopInd as follows:

$$\begin{array}{l} \mathsf{CopInd} :\ D_{\mathsf{CopInd}} \longrightarrow \mathbb{N}_0 \\ (\overline{P}(O_1), O_1, \ldots, \overline{P}(O_t), O_t) \\ \qquad \longmapsto \min \left\{ l \;\middle|\; \begin{array}{l} \exists i \neq j : \overline{P}_l(O'_i) = \overline{P}_l(O'_j) = \overline{P}_l(O_l), \\ \qquad \text{where } (\overline{P}_1(O'_1), O'_1, \ldots, \overline{P}(O'_{t+1}), O_{t+1}) \\ \qquad \leftarrow p\text{-}CLONE(\overline{P}_1(O_1), O_1, \ldots, \overline{P}(O_{t+1}), O_{t+1}) \end{array} \right\} \end{array}$$

Note that CopInd is defined in terms of the random variable $p\text{-}CLONE(\cdot)$. As $\min(\emptyset) = 0$, it can be seen rather easily that for the polynomial p and any objects $O_1, \ldots, O_{p(k)}$,

$$\begin{array}{l} \Pr\left[\ \mathsf{CopInd}(\overline{P}_1(O_1), O_1, \ldots, \overline{P}_{p(k)}(O_{p(k)}, O_{p(k)}) \neq 0\ \right] = \qquad (1) \\ \qquad = \Pr\left[\begin{array}{l} \exists\, i \neq j,\, l : \mathsf{Pro}(O'_i) = \mathsf{Pro}(O'_j) = \mathsf{Pro}(O_l), \\ \qquad \text{where } (\overline{P}'_1(O'_1), O'_1, \ldots, \overline{P}'_{p(k)+1}(O'_{p(k)+1}) \leftarrow p\text{-}CLONE(O_1, \ldots, O_{p(k)}) \end{array} \right] \end{array}$$

We can now calculate the success probability of $CLONE$ in the following way.

$$\Pr\left[\begin{array}{l} \overline{P}(O) = \overline{P}(O_1) = \overline{P}(O_2), \\ \qquad \text{where } (\overline{P}(O), O_1, O_2) \leftarrow \mathit{CLONE}(1^k, \overline{P}(O), O) \\ \qquad \text{and } O \leftarrow OG(1^k) \end{array}\right] =$$

$$= \Pr\left[\begin{array}{l} \mathsf{CopInd}(O_1, \ldots, O_{p(k)}) = i_0 \\ \qquad \text{where } i_0 \leftarrow_{u.a.r.} \{1, \ldots, p(k)\},\ O_{i_0} \leftarrow OG(1^k), \\ \qquad \text{and } O_i \leftarrow OG(1^k) \text{ for } i = 1, \ldots, i_0 - 1, i_0 + 1, \ldots, p(k) \end{array}\right]$$

$$= \Pr\left[\begin{array}{l} \mathsf{CopInd}(O_1, \ldots, O_{p(k)}) = i_0 \\ \qquad \text{where } i_0 \leftarrow_{u.a.r.} \{1, \ldots, p(k)\} \\ \qquad \text{and } O_i \leftarrow OG(1^k) \text{ for } i = 1, \ldots, p(k) \end{array}\right]$$

$$= \sum_{j=1}^{p(k)} \Pr\left[\begin{array}{l} \mathsf{CopInd}(O_1, \ldots, O_{p(k)}) = j \text{ and } i_0 = j \\ \qquad \text{where } i_0 \leftarrow_{u.a.r.} \{1, \ldots, p(k)\} \\ \qquad \text{and } O_i \leftarrow OG(1^k) \text{ for } i = 1, \ldots, p(k) \end{array}\right]$$

$$= \sum_{j=1}^{p(k)} \Pr\left[\begin{array}{l} i_0 = j \\ \qquad \text{where } i_0 \leftarrow_{u.a.r.} \{1, \ldots, p(k)\} \end{array}\right] \cdot$$

$$\cdot \Pr\left[\begin{array}{l} \mathsf{CopInd}(O_1, \ldots, O_{p(k)}) = j \\ \qquad \text{where } O_i \leftarrow OG(1^k) \text{ for } i = 1, \ldots, p(k) \end{array}\right]$$

$$= \frac{1}{p(k)} \cdot \sum_{j=1}^{p(k)} \Pr\left[\begin{array}{l} \mathsf{CopInd}(O_1, \ldots, O_{p(k)}) = j \\ \qquad \text{where } O_i \leftarrow OG(1^k) \text{ for } i = 1, \ldots, p(k) \end{array}\right]$$

$$= \frac{1}{p(k)} \cdot \Pr\left[\begin{array}{l} \mathsf{CopInd}(O_1, \ldots, O_{p(k)}) \neq 0 \\ \qquad \text{where } O_i \leftarrow OG(1^k) \text{ for } i = 1, \ldots, p(k) \end{array}\right]$$

$$= \frac{1}{p(k)} \cdot \Pr\left[\begin{array}{l} \exists\, i \neq j,\, l : \mathsf{Pro}(O'_i) = \mathsf{Pro}(O'_j) = \mathsf{Pro}(O_l), \\ \qquad \text{where } (O'_1, \ldots, O'_{p(k)+1}) \leftarrow p\text{-}\mathit{CLONE}(O_1, \ldots, O_{p(k)+1}) \\ \qquad \text{and } O_i \leftarrow OG(1^k) \text{ for } i = 1, \ldots, p(k) \end{array}\right]$$

Or, to summarize our calculation,

$$\Pr\left[\begin{array}{l}\overline{P}(O)=\overline{P}(O_1)=\overline{P}(O_2),\\ \qquad\text{where } (\overline{P}(O),O_1,O_2)\leftarrow \mathit{CLONE}(1^k,\overline{P}(O),O)\\ \qquad\text{and } O\leftarrow OG(1^k)\end{array}\right] = \tag{2}$$

$$= \frac{1}{p(k)}\cdot \Pr\left[\begin{array}{l}\exists\, i\neq j,\, l:\mathsf{Pro}(O'_i)=\mathsf{Pro}(O'_j)=\mathsf{Pro}(O_l),\\ \qquad\text{where } (O'_1,\ldots,O'_{p(k)+1})\leftarrow p\text{-}\mathit{CLONE}(O_1,\ldots,O_{p(k)+1})\\ \qquad\text{and } O_i\leftarrow OG(1^k) \text{ for } i=1,\ldots,p(k)\end{array}\right]$$

We further know from statement (1) that for infinitely many k and a polynomial q,

$$\Pr\left[\begin{array}{l}\exists\, i\neq j,\, l:\mathsf{Pro}(O'_i)=\mathsf{Pro}(O'_j)=\mathsf{Pro}(O_l),\\ \qquad\text{where } (O'_1,\ldots,O'_{p(k)+1})\leftarrow p\text{-}\mathit{CLONE}(O_1,\ldots,O_{p(k)+1})\\ \qquad\text{and } O_i\leftarrow OG(1^k) \text{ for } i=1,\ldots,p(k)\end{array}\right] > 1/q(k).$$

Inserting this into equation (2) we obtain that for infinitely many k a polynomial q and a polynomial q,

$$\Pr\left[\begin{array}{l}\overline{P}(O)=\overline{P}(O_1)=\overline{P}(O_2),\\ \qquad\text{where } (\overline{P}(O),O_1,O_2)\leftarrow \mathit{CLONE}(1^k,\overline{P}(O),O)\\ \qquad\text{and } O\leftarrow OG(1^k)\end{array}\right] > 1/p(k)\cdot 1/q(k).$$

Hence it holds that for infinitely many k and a polynomial $r \overset{\text{def}}{=} p\cdot q$,

$$\Pr\left[\begin{array}{l}\overline{P}(O)=\overline{P}(O_1)=\overline{P}(O_2),\\ \qquad\text{where } (\overline{P}(O),O_1,O_2)\leftarrow \mathit{CLONE}(1^k,\overline{P}(O),O)\\ \qquad\text{and } O\leftarrow OG(1^k)\end{array}\right] > 1/r(k).$$

This implies by the definition of unique object systems (Definition 5.2) that OS is no unique object system, which is at odds with the contradiction assumption. Therefore it provides the sought contradiction and completes the proof. $\square$

We are now in a position to prove the main theorem.

Theorem 6.10 (Main Theorem)**.** *Let $\mathcal{U}$ be a universe, and $\mathcal{T}$ be a technology in that universe. Let DSS be a φ-secure signature scheme in $\mathcal{T}$, and let OS be a unique object system in $\mathcal{T}$. Then, the standard labeling scheme from DSS and OS, $\mathsf{SL}(OS, DSS)$, is a secure labeling scheme in $\mathcal{T}$.*

Proof. We lead the proof by contradiction, assuming that DSS is a φ-secure signature scheme in $\mathcal{T}$ and that OS is a unique object system in $\mathcal{T}$, but that $\mathsf{SL}(OS, DSS)$ is not a secure labeling system in $\mathcal{T}$. By use of lemma 6.9 this is equivalent to the following contradiction assumption:

Contradiction Assumption: *DSS* is a φ-secure signature scheme in $\mathcal{T}$, *OS* is a p-UOS in $\mathcal{T}$, and $LS \overset{\text{def}}{=} SL(OS, DSS)$ is not a secure labeling scheme in $\mathcal{T}$.

The assumption that *LS* is no secure labeling scheme in $\mathcal{T}$ implies the following:

Statement (1): There is a probabilistic polynomial PhTM *FAKE* in $\mathcal{T}$ and polynomials p, q such that for infinitely many n,

$$\Pr\left[\begin{array}{l} T(t, L'_i) = 1 \quad \text{for } i = 1, \ldots, p(n)+1 \\ \qquad \text{where } (L'_1, \ldots, L'_{p(n)+1}) \leftarrow \mathit{FAKE}(L_1, \ldots, L_{p(n)}), \\ \qquad L_1 \leftarrow LG(s),\ \ldots,\ L_{p(n)} \leftarrow LG(s) \quad \text{and} \quad (s, v) \leftarrow C(1^n) \end{array}\right] \geq q(n),$$

where the probability is taken over the random outputs of C, P, T and *FAKE*.

A short outline of the further proof is as follows. Lemma 6.7 tells us that if *FAKE* is successful, then there is either a cloned object or a new digital signature among the faked labels $L_1, \ldots, L_{p(n)+1}$. Hence, searching the output of *FAKE* for a cloned object or a new signature will enable us to "break" the signature scheme *DSS* or the p-unique object system *OS*. If either of them is broken and hence insecure, however, then we are at odds with the contradiction assumption, which provides a contradiction and completes the proof.

Still, the formal realization of this argument requires considerable technical effort. One reason is that Lemma 6.7 only speaks about one single output of *FAKE*. Contrary to that, the security definitions of signature schemes and unique object systems are asymptotic, whence we have to consider infinitely many outputs. The other reason is that the adversarial models for φ-secure signature schemes and unique object systems differ. In the case of signature schemes the adversary may act adaptively: It can choose the newly queried signatures in dependence of the signatures queried earlier. This setting enforces that the attacker is modelled as an *oracle* probabilistic PhTM. In opposition to that, the input for an attack on a unique object systems is chosen non-adaptively uniformly at random, whence the attacker is modelled as a *normal* PhTM. Therefore, the earlier idea to let *one single* machine search the output of *FAKE* and to let this machine either break the signature scheme or the unique object system – depending on the output of *FAKE* – will not work. Such a machine would either fail to meet the formal attack model for signature schemes or the attack model for unique object system.

Hence, we will construct two separate machines *SIGBR* and *CLONE* instead (one of them equipped with an oracle, the other one not). *SIGBR* will be equipped with a signature oracle and will try to break the φ-security of the signature scheme *DSS*. *CLONE* will have no signature oracle and will try to break the unique object system. The sought contradiction will be reached if we can infer that under the premise of statement (1), one of the machines must be successful with significant probability. This can be achieved by application of Lemma 6.7 and some further probability analysis of *SIGBR* and *CLONE*.

We will now formally introduce the machines *SIGBR* and *CLONE*, starting with the latter.

```
MACHINE $CLONE$:
    Input $\left(1^k, \overline{P}(O_1), O_1, \ldots, \overline{P}(O_{p(k)}), O_{p(k)}\right)$
        Set $(s, v) \leftarrow G(1^k)$
        Set $S_i \leftarrow Sig(s, \overline{P}(O_i))$ for $i = 1, \ldots, p(k)$
        Set $L_i \leftarrow (\overline{P}(O_i), S_i, O_i)$ for $i = 1, \ldots, p(k)$
        Set $(L'_1, \ldots, L'_{p(k)+1}) \leftarrow FAKE\ (L_1, \ldots, L_{p(k)})$
        If $\{\mathsf{Pro}(L_i) \mid i = 1, \ldots, p(k)\} = \{\mathsf{Pro}(L'_i) \mid i = 1, \ldots, p(k)+1\}$
            then proceed, else output "failure" and abort.
    Output $\left(\mathsf{Obj}(L'_1), \ldots, \mathsf{Obj}(L'_{p(k)+1})\right)$
```

CLONE in effect does the following: It takes as input $p(k)$ objects $O_1, \ldots, O_{p(k)}$ and tries to produce a copy of one of these objects (recall that we are considering p-unique object systems by virtue of Lemma 6.9). To that purpose, it wants to utilize the PhTM *FAKE* trying to feed the objects O_i into *FAKE* in order to be copied. The problem is, however, that *FAKE* takes labels, not objects as inputs. *CLONE* hence has to turn the objects O_i into labels L_i. This can be done by imitating the label generation process of the standard labeling scheme: *CLONE* simply generates a signature key at random and produces signatures for the properties $\overline{P}(O_i)$ of the objects O_i. Then, it sets the labels $L_i \overset{\text{def}}{=} (\overline{P}(O_i), Sig(s, \overline{P}(O_i), O_i)$ and feeds the generated labels into *FAKE*. In return, *FAKE* outputs $k+1$ labels $L'_1, \ldots, L'_{k+1}$.

If they are all valid labels, then we know by Lemma 6.7 that either one new signature for a new object has been produced by *FAKE*, which does not help the algorithm *CLONE*, and it outputs a failure notice. Or, one of the objects $O_1, \ldots, O_k$ has been copied. This *does* help the algorithm *CLONE*, and it outputs that object together with its copy.

The machine *SIGBR* whose aim is to break the signature scheme by utilizing *FAKE* can be constructed along similar lines. Contrary to *CLONE*, however, *SIGBR* hopes that the output of *FAKE* contains a faked signature, not a cloned object. If that is the case, *SIGBR* can output a faked signature; otherwise, it outputs a failure notice.

According to the adaptive attack scenario for φ-secure signature schemes, *SIGBR* is provided with a signing oracle S_s. For any queried string x this oracle returns the string $SO_s(x)$, where $SO_s(x) \overset{\text{def}}{=} Sig(s,x)$, where Sig is the signing algorithm of DSS and s is the corresponding signing key.

```
MACHINE $SIGBR$:

  Input  $(1^k, v)$
      Set  $(\overline{P}_i(O_i), O_i) \leftarrow OG(1^k)$   for $i = 1, \ldots, p(n)$
      Set  $S_i \leftarrow SO_s(\overline{P}_i(O_i))$   for $i = 1, \ldots, p(n)$
      Set  $L_i \leftarrow (\overline{P}_i(O_i), S_i, O_i)$   for $i = 1, \ldots, p(n)$
      Set  $(L'_1, \ldots, L'_{p(n)+1}) \leftarrow FAKE(L_1, \ldots, L_{p(n)})$
      If  $\exists\, i_0 \in \{1, \ldots, p(n)+1\} : \begin{pmatrix} Ver\,(v, \mathsf{Pro}(L'_{i_0}), \mathsf{Sig}(L'_{i_0})) = 1 \\ \text{and } \nexists\, j \in 1, \ldots, p(n) : \mathsf{Pro}(L'_{i_0}) = \mathsf{Pro}(L_j) \end{pmatrix}$
          then  proceed, else output “failure” and abort
  Output  $(\mathsf{Pro}(L'_{i_0}), \mathsf{Sig}(L'_{i_0}))$
```

We will now analyse the success probabilities of $SIGBR$ and $CLONE$ in dependency of the success probability of $FAKE$. It holds for any $k \in \mathbb{N}$ that

$$\Pr\left[\begin{array}{l} T(t, L'_i) = 1 \quad \text{for } i = 1, \ldots, p(k)+1 \\ \qquad \text{where } (L'_1, \ldots, L'_{p(k)+1}) \leftarrow FAKE(L_1, \ldots, L_{p(k)}), \\ \qquad L_1 \leftarrow LG(s),\ \ldots,\ L_{p(k)} \leftarrow LG(s) \text{ and } (s,v) \leftarrow C(1^k) \end{array}\right]$$

$$= \Pr\left[\begin{array}{l} T(t, L'_i) = 1 \quad \text{for } i = 1, \ldots, p(k)+1 \\ \text{and } T(t, L_i) = 1 \quad \text{for } i = 1, \ldots, p(k) \\ \qquad \text{where } (L'_1, \ldots, L'_{p(n)+1}) \leftarrow FAKE(L_1, \ldots, L_{p(k)}), \\ \qquad L_1 \leftarrow LG(s),\ \ldots,\ L_{p(k)} \leftarrow LG(s) \text{ and } (s,v) \leftarrow C(1^k) \end{array}\right]$$

$$= \Pr\left[\begin{array}{l} \begin{pmatrix} \exists\, i \neq j,\, l:\ M\big(\mathsf{Par}(L_l), \mathsf{Obj}(L'_i)\big) = \\ \quad = M\left(\mathsf{Par}(L_l), \mathsf{Obj}(L'_j)\right) = M\big(\mathsf{Par}(L_l), \mathsf{Obj}(L_l)\big) \end{pmatrix} \\ \text{or } \begin{pmatrix} \exists i:\ Ver\big((v, \mathsf{Pro}(L'_i), \mathsf{Sig}(L'_i)\big) = 1 \\ \text{and } \nexists j : \mathsf{Pro}(L'_i) = \mathsf{Pro}(L_j) \end{pmatrix} \\ \text{where } (L'_1, \ldots, L'_{p(k)+1}) \leftarrow FAKE(L_1, \ldots, L_{p(k)}), \\ L_1 \leftarrow LG(s),\ \ldots\ ,\ L_{p(k)} \leftarrow LG(s) \text{ and } (s,v) \leftarrow C(1^k) \end{array}\right]$$

(because of Lemma 6.7)

$$\leq \ \Pr\left[\begin{array}{l} \exists\, i \neq j,\, l:\ M\left(\mathsf{Par}(L_l), \mathsf{Obj}(L'_i)\right) = \\ \quad = M\left(\mathsf{Par}(L_l), \mathsf{Obj}(L'_j)\right) = M\left(\mathsf{Par}(L_l), \mathsf{Obj}(L_l)\right) \\ \qquad \text{where } (L'_1, \ldots, L'_{p(k)+1}) \leftarrow \mathit{FAKE}(L_1, \ldots, L_{p(k)}), \\ \qquad L_1 \leftarrow \mathit{LG}(s),\ \ldots,\ L_{p(k)} \leftarrow \mathit{LG}(s) \text{ and } (s,v) \leftarrow C(1^k) \end{array}\right]$$

$$+ \ \Pr\left[\begin{array}{l} \exists i:\ \mathit{Ver}\big((v, \mathsf{Pro}(L'_i), \mathsf{Sig}(L'_i)\big) = 1 \\ \quad \text{and } \nexists j : \mathsf{Pro}(L'_i) = \mathsf{Pro}(L_j) \\ \qquad \text{where } (L'_1, \ldots, L'_{p(k)+1}) \leftarrow \mathit{FAKE}(L_1, \ldots, L_{p(k)}), \\ \qquad L_1 \leftarrow \mathit{LG}(s),\ \ldots,\ L_{p(k)} \leftarrow \mathit{LG}(s) \text{ and } (s,v) \leftarrow C(1^k) \end{array}\right]$$

(because $P(A \cup B) \leq P(A) + P(B)$)

$$= \ \Pr\left[\begin{array}{l} \mathit{CLONE}\left(1^k, \overline{P}(O_1), O_1, \ldots, \overline{P}(O_{p(k)}), O_{p(k)}\right) \neq \text{“}failure''\\ \qquad \text{where } O_i \leftarrow \mathit{OG}(1^k) \quad \text{for } i = 1, \ldots, p(k) \end{array}\right]$$

$$+ \ \Pr\left[\begin{array}{l} \mathit{SIGBR}\,(1^k, v) \neq \text{“}failure'' \\ \qquad \text{where } (s,v) \leftarrow C(1^k) \end{array}\right]$$

(because of the design of *SIGBR* and *CLONE*)

Hence, we obtain from statement (1) that there is a polynomial q such that for infinitely many k,

$$\Pr\left[\begin{array}{l} \mathit{CLONE}\left(1^k, \overline{P}(O_1), O_1, \ldots, \overline{P}(O_{p(k)}), O_{p(k)}\right) \neq \text{“}failure''\\ \qquad \text{where } O_i \leftarrow \mathit{OG}(1^k) \quad \text{for } i = 1, \ldots, p(k) \end{array}\right]$$

$$+ \ \Pr\left[\begin{array}{l} \mathit{SIGBR}\,(1^k, v) \neq \text{“}failure'' \\ \qquad \text{where } (s,v) \leftarrow C(1^k) \end{array}\right]$$

$$\geq \ \Pr\left[\begin{array}{l} T(t, L'_i) = 1 \quad \text{for } i = 1, \ldots, p(k)+1 \\ \qquad \text{where } (L'_1, \ldots, L'_{p(k)+1}) \leftarrow \mathit{FAKE}(L_1, \ldots, L_{p(k)}), \\ \qquad L_1 \leftarrow \mathit{LG}(s),\ \ldots,\ L_{p(k)} \leftarrow \mathit{LG}(s) \text{ and } (s,v) \leftarrow C(1^k) \end{array}\right]$$

(by the previous calculation)

$$> \ 1/q(k)$$

(by statement (1)).

This implies that for infinitely many k,

$$\Pr\left[\begin{array}{c} \mathit{CLONE}(1^k, \overline{P}(O_1), O_1, \ldots, \overline{P}(O_{p(k)}, O_{p(k)}) \neq \text{“}\mathit{failure}\text{''} \\ \text{where } O_i \leftarrow OG(1^k) \quad \text{for } i = 1, \ldots, p(k) \end{array}\right] > 1/q(k)$$

or

$$\Pr\left[\begin{array}{c} \mathit{SIGBR}(1^k, v) \neq \text{“}\mathit{failure}\text{''} \\ \text{where } (s, v) \leftarrow C(1^k) \end{array}\right] > 1/q(k).$$

Therefore we obtain by the definitions of φ-secure signature schemes and p-unique object systems (Def. 4.5 and 6.8) that OS is no p-unique object system or DSS is no secure signature scheme.

This is at odds with the contradiction assumption, which states that both OS is a p-unique object system and DSS is a secure signature scheme. Hence it provides the sought contradiction and completes the proof. □

7 Summary

We introduced Physical Turing Machines (or PhTMs, for short) as a new formal machine model in this paper, and discussed their applicability to the formal treatment of classical and of physical cryptography.

We first suggested that PhTMs could be a suitable tool to formalize classical cryptography in such a way that physical computations by the adversary (quantum, optical, etc.) and physical attacks like side channels could be included. In this context, we sketched a few new, adjusted definitions in order to illustrate our point, but kept this part of our treatment relatively short, since it was not our main topic in this paper. Future efforts will have to work out the application of PhTMs in this area in full detail.

A topic we addressed in greater detail, and which actually was the main topic of this paper, was the formalization of physical cryptography by use of PhTMs. In particular, we exemplarily formalized a standard scheme from physical cryptography, which concerns the generation of forgery proof physical labels (tags/markers) by use of so-called unique objects in combination with digital signature schemes (compare [10] and references therein). This example scheme was chosen by us for a number of reasons: First of all, it is very intuitive and can be understood without a strong background in PUFs. Secondly, the necessity of the physical unclonability of the used unique object is obvious in the scheme (perhaps yet more than in related PUF-based schemes), and so is the need to formalize this unclonability. Thirdly, the scheme is a hybrid scheme in the sense that it combines a classical, complexity based tool (namely digital signatures) with the physical feature of unclonability. The latter makes a formal treatment particularly difficult, since the used machine model must be able to capture both classical, asymptotic, computational aspects as well as physical aspects.

We introduced PhTMs as a solution of this problem. We showed that they can be used for formally defining the relevant notions of a unique object system and a labeling scheme, and for leading a formal reductionist security proof. PhTMs provide the machine model for this proof, and, if you like, serve as its "formal backbone". Our proof shows that the security of the labeling scheme can be reduced to the assumptions that the employed digital signature scheme and unique object system are secure.

One central further ingredient in our model besides PhTMs is the concept of a *"technology"*. A technology is a set of functions that maps numbers and objects to numbers and objects, and which subsumes the current state of human technology and craftsmanship at a given point in time. The use of *technologies* in our model can help us to find a balance between the following two extremes: (i) Allowing currently unrealistic actions and computations (such as practically allowing all theoretically feasible quantum computations, which would allow for factoring arbitrarily large numbers in polynomial time), and (ii) ignoring all physical actions and physical computations in the formalization of cryptography. It turns out that it is not necessary in a *reductionist* proof to exactly specify the technology. PhTM nevertheless, i.e. without such a specification, allow proofs of statements of the following form: If scheme A is secure against attacks with current technology, and scheme B is secure against attacks with current technology, then so is scheme C (which is built from A and B).

Another noteworthy aspect of the presented model of PhTMs is their asymptotic character. While the objects used in physical cryptography (such as physical unclonable functions (PUFs) or unique objects (UNOs)) are finite physical systems with a finite number of atoms and a finite number of input–output pairs, the traditional treatment of cryptographic security is based on inherently asymptotic concepts, such as polynomial time and negligible probability. Reconciling these two finite and infinite worlds can be difficult; it has led to formal problems in several early PUF definitions, which have been discussed in a number of publications on the foundations of PUFs [31, 25].

Future Work. Several suggestive lines of future work arise from the presented material. One example would be to further investigate how PhTMs could be applied to formalize the security of hardware implementation of classical cryptographic schemes, for example against invasive and side channel attacks. Due to their asymptotic nature, PhTMs could probably reconcile the gap between a finite physical system/implementation on the one hand, and the traditionally asymptotic security notions of cryptographic schemes on the other hand. They might allow us to lead reductionist proof for the general security of hardware tokens that could both span over the mathematical security of the implemented scheme and the physical security of the hardware implementation.

Another natural step would be the formalization of further schemes from physical cryptography by PhTMs. Examples could be several recent protocols for PUFs and related primitives, including identification, message authentication, key exchange, or oblivious transfer (compare [25, 24, 5, 21]). PhTMs appear very useful to this end.

References

[1] Scott Aaronson: *NP-complete Problems and Physical Reality.* Electronic Colloquium on Computational Complexity (ECCC), 026, 2005.

[2] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, François-Xavier Standaert, Christian Wachsmann: *A Formalization of the Security Features of Physical Functions.* IEEE Symposium on Security and Privacy 2011, pp. 397-412.

[3] Nathan Beckmann, Miodrag Potkonjak: *Hardware-Based Public-Key Cryptography with Public Physically Unclonable Functions.* Information Hiding 2009, pp. 206-220.

[4] Charles H. Bennett, Gilles Brassard: *Quantum cryptography: Public key distribution and coin tossing.* Proceedings of IEEE International Conference on Computers, Systems and Signal Processing. Vol. 175, No. 150, 1984.

[5] Christina Brzuska, Marc Fischlin, Heike Schröder, Stefan Katzenbeisser: *Physically Uncloneable Functions in the Universal Composition Framework.* CRYPTO 2011, pp. 51-70, 2011.

[6] James D. R. Buchanan et al: *Fingerprinting documents and packaging.* Nature 436.28 (2005): 475.

[7] Johannes Buchmann et al: *Post-Quantum Signatures.* IACR Cryptology ePrint Archive 2004 (2004): 297.

[8] Jie Chen: *A DNA-based, biomolecular cryptography design.* ISCAS 2003.

[9] Catherine Taylor Clelland, Viviana Risca, Carter Bancroft: *Hiding messages in DNA microdots.* Nature 399 (6736), pp. 533-534, 1999.

[10] Gerald DeJean, Darko Kirovski: *RF-DNA: Radio-Frequency Certificates of Authenticity.* CHES 2007, pp. 346-363.

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

[12] Blaise Gassend, Dwaine E. Clarke, Marten van Dijk, Srinivas Devadas: *Silicon physical random functions.* ACM Conference on Computer and Communications Security 2002, pp. 148-160.

[13] Ashish Gehani, Thomas LaBean, John Reif: *DNA-based cryptography.* In: Aspects of Molecular Computing. Springer Berlin Heidelberg, 2004, pp. 167-188.

[14] Darko Kirovski: *Anti-counterfeiting: Mixing the physical and the digital world.* In: Towards Hardware-Intrinsic Security. Springer, 2010, pp. 223-233.

[15] Oded Goldreich: *The Foundations of Cryptography - Volume 2, Basic Applications.* Cambridge University Press 2004, ISBN 0-521-83084-2.

[16] Shafi Goldwasser, Silvio Micali, Ronald L. Rivest: *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks.* SIAM J. Comput. 17(2): 281-308 (1988)

[17] Arjen K. Lenstra, Adi Shamir: *Analysis and optimization of the TWINKLE factoring device.* EUROCRYPT 2000, pp. 35-52.

[18] Ahmed Mahmoud, Ulrich Rührmair, Mehrdad Majzoobi, Farinaz Koushanfar: *Combined Modeling and Side Channel Attacks on Strong PUFs.* IACR Cryptology ePrint Archive 2013: 632 (2013)

[19] Ueli M. Maurer: *Conditionally-perfect secrecy and a provably-secure randomized cipher.* Journal of Cryptology 5.1 (1992): 53-66.

[20] Ueli M. Maurer: *Secret key agreement by public discussion from common information.* Information Theory, IEEE Transactions on 39.3 (1993): 733-742.

[21] Rafail Ostrovsky, Alessandra Scafuro, Ivan Visconti, Akshay Wadia: *Universally Composable Secure Computation with (Malicious) Physically Uncloneable Functions.* EUROCRYPT 2013, pp. 702-718.

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

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

[24] Ulrich Rührmair: *Oblivious Transfer Based on Physical Unclonable Functions.* TRUST 2010, pp. 430-440.

[25] Ulrich Rührmair, Heike Busch, Stefan Katzenbeisser: *Strong PUFs: Models, Constructions, and Security Proofs.* In: Towards Hardware-Intrinsic Security, Springer, 2010, pp. 79-96.

[26] Ulrich Rührmair, Srinivas Devadas, Farinaz Koushanfar: *Security based on Physical Unclonability and Disorder.* In M. Tehranipoor and C. Wang (Editors): Introduction to Hardware Security and Trust. Springer, 2011

[27] Ulrich Rührmair, Marten van Dijk: *PUFs in Security Protocols: Attack Models and Security Evaluations.* IEEE Symposium on Security and Privacy 2013, pp. 286-300.

[28] U. Rührmair, C. Jaeger, C. Hilgers, M. Algasinger, G. Csaba, M. Stutzmann: *Security Applications of Diodes with Unique Current-Voltage Characteristics.* 14th International Conference on Financial Cryptography and Data Security (FC 2010). Lecture Notes in Computer Science, Volume 6052, Springer, 2010.

[29] U. Rührmair, C. Jaeger, M. Bator, M. Stutzmann, P. Lugli, G. Csaba: *Applications of High-Capacity Crossbar Memories in Cryptography.* IEEE Transactions on Nanotechnology 10(3), pp. 489-498, 2011.

[30] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, Jürgen Schmidhuber: *Modeling attacks on physical unclonable functions*. ACM Conference on Computer and Communications Security 2010, pp. 237-249, 2010.

[31] Ulrich Rührmair, Jan Sölter, Frank Sehnke: *On the Foundations of Physical Unclonable Functions*. IACR Cryptology ePrint Archive, Report 2009/277, 2009.

[32] Ulrich Rührmair, Xiaolin Xu, Jan Sölter, Ahmed Mahmoud, Farinaz Koushanfar, Wayne Burleson: *Power and Timing Side Channels for PUFs and their Efficient Exploitation*. IACR Cryptology ePrint Archive, Report 2013/851, 2013.

[33] Ulrich Rührmair, Xiaolin Xu, Jan Sölter, Ahmed Mahmoud, Farinaz Koushanfar, Wayne Burleson: *Efficient Power and Timing Side Channels for Physical Unclonable Functions*. CHES 2014, to appear.

[34] Adi Shamir: *"Factoring large numbers with the TWINKLE device."* CHES 2009, pp. 2-12.

[35] Peter W. Shor: *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*. SIAM journal on computing 26.5, pp. 1484-1509, 1997.

[36] Andrew Chi-Chih Yao: *Classical physics and the Church-Turing Thesis*. Journal of the ACM 50(1), 100-105, 2003.

Part IV

Summary and Future Research

This thesis investigated the foundations of physical unclonable functions (PUFs) and related primitives. Laying these solidly appears indispensible for a sound, long-term development of the field, and therefore constitutes a central research task for the entire area of PUFs. We thereby made novel contributions to the following general research strands:

- *Foundational work*, including the classification and demarcation of different, PUF-related primitives and concepts.
- *Fundamental usability of PUFs in cryptographic protocols*, including the theoretical potential of PUFs in such protocols, the correct attack models for practical PUF appliances, and practical security analyses of existing PUF protocols.
- *Mathematical formalization*, including formal mathematical definitions of PUF-related primitives, and the conductance of formal security proofs based on these definitions.

In greater detail, our thesis contains the following original material:

- With respect to *foundational work*, we were the first to dedicatedly and explicitly investigate the formal foundations of PUFs in 2008 and 2009 [45, 74], and to semi-formally classify different PUF-primitives and their security properties in 2009 and 2011 [74, 58]. These efforts have mainly been reported in Chapters 2 and 7.
  - One of our main achievements there was to formally differentiate between UNOs, Weak PUFs, Strong PUFs, SHIC PUFs, and other disorder-based primitives. We developed semi-formal specifications for all of them and worked out archetypical application scenarios of theirs.
  - Furthermore, we took a second look at previous PUF-definitions, noting some problematic aspects. Some of them related to the use of asymptotic concepts like polynomial time in connection with single PUFs, which necessarily only have a finite challenge-response space.
  - We were the first to define the security of PUFs as a *game* between the PUF and the adversary. This application of game-theoretic concepts in the area of PUFs has been picked up by a large number of follow-up papers at major conferences, including Armknecht et al. at IEEE S&P 2011 [2].
  - We first observed that for fundamental physical reasons, the maximal entropy in a PUF can be at most polynomial in the size of the PUF. As a consequence, constructing information-theoretically secure PUFs with an exponential number of challenges is impossible. Besides other things, this implies that Strong PUFs should ideally be modeled with computational definitions, not with information-theoretical ones. This observation is highly relevant for some recent PUF-definitions and proofs, all of which, unfortunately, have been lead in an information-theoretic framework [6, 40].

- In a second strand of original work, we dealt with the *fundamental usability of PUFs in cryptographic protocols*. Among other things, we investigated the theoretical and practical usability of PUFs in protocols like oblivious transfer, bit commitment, or key exchange. We thereby made the following new contributions:

  - We were the first to show that so-called *"Strong PUFs"* have the potential to act as a universal cryptographic primitive. More precisely, we proved that Strong PUFs can implement oblivious transfer, at least in isolated, stand-alone settings where the PUF is used only once, and where the PUF is non-manipulated, possessesing only the expected security features. This observation paved the way for several follow-up works, including papers at CRYPTO 2011 [6] and EUROCRYPT 2013 [40]. It opened up a new interpretation of Strong PUFs as a powerful cryptographic primitive, in opposition to the predominant earlier understanding of (Weak) PUFs as a mere tool for secure key storage.
  - We unocvered that quadratic attacks exist on two oblivious transfer and bit commitment protocols published by Brzuska, Fischlin, Schröder and Katzenbeisser at CRYPTO 2011 [6]. This surprisingly makes the protocols practically insecure when they are implemented in connection with optical PUFs, as explicitly suggested at CRYPTO 2011 [6]. The attacks are very effective can be mounted with very little effort in practice. Our work illustrated that finite parameters *do* matter when PUF schemes are realized in hardware, and arguably lead to a new understanding of PUF security: Asymptotic concepts are likely not the right formalism to express PUF security features.
  - We first revealed vulnerabilities in a session key exchange protocol by Tuyls and Skoric [86], which is run between a bank and a bank card containing the PUF. The protocol becomes insecure if adversaries have access to the PUF on the card several times, an assumption that is highly realistic in a setting where the bank is re-used in several terminals. Earlier session keys can be derived under this assumption.
  - We abstracted from the abovementioned analysis of Tuyls' and Skoric's protocol [86], and introduced two new, general, and very fundamental attack models on PUF protocols: The so-called *"bad PUF model"* and the *"PUF re-use model"*. Again, these have led to strong follow up works in the community at top conferences, e.g., Damgard and Scafuro at ASIACRYPT 2013 [17], or Dachman-Soled, Fleischhacker, Katz, Lysyanskaya, and Schröder at CRYPTO 2014 [16].
  - We subsequently showed that protocols of Brzuska, Fischlin, Schröder and Katzenbeisser at CRYPTO 2011 [6] and partly the protocols of Ostrovsky, Scafuro, Visconti and Wadia at EUROCRYPT 2013 [40] are not secure in the PUF re-use model and the bad PUF model. This delimits their usability in practical and economically viable applications, especially in situation

where the PUF shall be used more than once, and shall not have to be disposed and destroyed already after a single use.

  - As hardware countermeasures, we formally introduced two novel PUF concepts called *"Erasable PUFs"* and *"Certifiable PUFs"*. We took first steps in hardware experiments and simulations to the realization of Erasable PUFs via the use of so-called nano-scale crossbar architectures. Our construction is able to establish information-theoretic security.

- Regarding the *mathematical formalization* of PUFs, we were the first to develop sound formal definitions for so-called *"Unique Objects"* (UNOs) in 2011 [48]. The corresponding material has been given in Chapter 8. In greater detail, it makes the following novel contributions:

  - We extended existing approaches in PUF-formalization, for the first time considering adversaries who execute arbitrary physical actions on PUFs and UNOs, not just standard CRP-measurements. To model such adversaries, we introduced a new formal computational model, so-called *"physical Turing machines"*, and the concept of a *"technology"*, which stipulates the admissible physical actions of the adversary.
  - We used physical Turing machines and technologies to comprehensively formalize and define UNOs as well as a special application of theirs, namely the secure labeling of valuable objects and items (like branded products, passports, banknotes, and the like).
  - We led the first formal proof for UNOs, and the first security proof in the general PUF-area that explicitly allows general physical attacks of the adversary. Our proof applies to arbitrary adversaries who are only limited by the current state of technology in their (physical) actions, and who are not a priori bound to using the CRP-interface of the PUF or UNO, respectively. It thereby transfers for the first time classical, reductionist techniques into our new, physical context.

  We believe that physical Turing machines and the associated concept of a *"technology"* will likely have applications beyond PUFs and UNOs, for example in the formalization of arbitrary physical security features, or also in complexity theory. They might be among the most fruitful new concepts introduced in this thesis.

The seven papers of this cumulative thesis have partly been published at the top venues of the community, including CHES 2012 [59] and the IEEE Symposium on Security and Privacy 2013 [60], with the latter having an acceptance rate of merely 12%. The paper at CHES 2012 was named as one of the best papers of the conference, and selected for a special issue of the Journal of Cryptographic Engineering in 2013 [61].

All papers employed in this cumulative thesis in sum have been quoted around 330 times until September 26, 2016 according to Google Scholar (with the candidate being first author or sole author on all of them) [26].

Future Research. We would like to conclude this thesis by a discussion of future research opportunities — not just regarding the foundations of PUFs, but concerning the wider area of PUF-research. In our opinion, the most promising fields for future investigations will likely include the following:

- *Foundations, Formalization, Classification:* A continuing formalization and consolidation of the area remains vital for its sound future development. Among other things, this must include an ongoing, refined specifications/definitions for existing and future PUFs types, the identification of typical applications for these different PUF types, as well as formal security proofs. Interestingly, such a sound formal specification indeed seems necessary for an efficient communication within the multi-disciplinary PUF research teams: It will help to accurately specify the tasks of the electrical engineers implementing the PUFs, as well as those of the computer scientists and system designers that use PUFs in concrete protocols. This area will hence remain very active in the future, building a bridge from PUFs to the theoretical cryptography community.

- *Attacks and Countermeasures:* The large number of recent, successful attacks has shown that PUFs are no magic toolbox that can miraculously create security for free. Extending the existing hardware and protocol attacks, and possibly creating entirely new ones, will hence continue to be a prospering topic. The same holds for the development and implementation of effective countermeasures.

- *Improved, Optimized, or Novel Implementations:* The general concept of PUFs can only shine if suitable, i.e., secure and efficient, hardware realizations are developed. Strong, continuous potential therefore lies in the improvement and optimization of existing implementations, and in the development of new ones. Optimizable parameters include costs, stability, PUF area and size, power consumption, and security, among others. This subfield can be expected to be extremely active over the upcoming years.

- *Applications of Nanotechnology and Nanomaterials:* Over the last decades, both nanotechnology and security have been two of the most successful subdisciplines within the sciences. PUFs lie exactly at their intersection. It seems very likely that the already emerging merger of these two fields will continuously gain momentum. It will almost certainly create entirely new reserach opportunities that substantially exceed the current topics in the area. Examples include new implementations of known PUF types by nanotechnology and nanomaterials, dedicated design of nano-materials with certain security properties, or implementation of novel disorder-based primitives like SIMPL systems [46] by nano-techniques.

- *New Uses of Physical Disorder in Security, and new Disorder-Based Primitives:* Besides PUFs and related methods, there seems to be significant potential for new, currently unseen uses of physical disorder in security. Any new discovery will spark new research, keeping the field prospering. One recent approach in this direction might be the Virtual Proofs of Reality of the candidate [54, 68] and the use of PUF-sensors put forward by Rosenfeld et al. [44], but there are certainly many others.

- *Commercialization:* One last subfield where strong activity can be expected is the commercialization of PUFs. What are their exact use cases and unique selling propositions compared to classical techniques that PUFs and related primitives have to offer? What are the costs of adapting existing production lines? Do the expected advantages outweigh these costs? How can the transfer of PUFs from scientific theory to commercial applications be accomplished safely, i.e., without unwanted security gaps? Since it seems hard to imagine that the current interest in PUFs will persist over long periods without any solid commercial use cases, finding satisfactory answers to these questions is central to the entire area, and also affects academic PUF research in general.

Following these and other questions, we believe that the field will continue to expand and flourish in the foreseeable future. The fact that it spans from theoretical computer science over electrical engineering to nanophysics gives it a very special appeal, drawing researchers from various different disciplines into the area. Slightly over ten years after their introduction, this currently makes PUFs one of the most thrilling sub-topics in security and cryptographic research, a trend that will most likely continue and even further intensify in the foreseeable future.

Acknowledgements

I would like to coordially thank my supervisor at TU Berlin, Prof. Dr. Jean-Pierre Seifert, who guided this thesis in its final phase with all his enthusiasm, inspiration and support. Nothing less can be said about the other members of my PhD committee: Prof. Dr. Marten van Dijk from the University of Connecticut and MIT, a close collaborator and friend over the last years, MIT's Prof. Srinivas Devadas, PhD, one of the highly esteemed founding fathers of the entire area. I feel honored and privileged to have them on my PhD committee. In an often highly hierarchical university system, they are members of a rare species: They embody Humboldt's ideal of a relationship between professor and student, where neither the teacher should serve the pupil, nor the pupil serve the teacher, but both serve science [31]. The common enthusiasm for the subject, so Humboldt envisioned, would blur hierarchical borderlines, leading to the inspirational atmosphere that enables high ranking research (in addition to making it enjoyable). It seems to me that two centuries after its original formulation, Humboldt's vision continues to be of relevance, and can still serve as inspiration for a fruitful university life.

I am also deeply grateful to many other colleagues, teachers and scientists who crossed my path and inspired me over all the years: Wayne Burleson, Peter Clote, Josef Friedrich, Dan Holcomb, Max Jakob, Jan Johannsen, Stefan Katzenbeisser, Darko Kirovski, Dima Kononchuk, Sven Kosub, Farinaz Koushanfar, Mike Mosca, Andreas Pfitzmann, Helmut Schwichtenberg, Jan Sölter, Gabrielle Stoy, Dominic Welsh, Jürgen Wild, Stefan Wolf, Jürg Wullschleger, Xiaolin Xu, and Andreas Zumbusch (in alphabetical order) are all part of this necessarily incomplete list. Thanks to all of them for all of their inspiration, their ideas, and their support! Special thanks goes to Jana Peich and Peter Marock from TU Berlin and Wendy Williams and Fiona Spensley from the University of Oxford. I wish all university administratives were like them!

More than to anyone else, however, I am indebted to my parents, grandparents and family. The statement that without them, this thesis would never have been possible, holds true in almost indefinitely many ways. Without their love, support and encouragement, it could never have been what it is today.

Thanks to them for the journey they gave me!

Part V

Appendix

Appendix A

Complete Publication List

Not all publications of the candidate have been used in this thesis. A comprehensive list of his scientific publications to this date is therefore given below (status: September 26, 2016). The list is ordered chronologically and by publication medium.

Journals:

1. C. Jaeger, M. Algasinger, U. Rührmair, G. Csaba, M. Stutzmann: *Random p-n-junctions for physical cryptography.* Applied Physics Letters 96, 172103, 2010.

2. U. Rührmair, C. Jaeger, M. Bator, M. Stutzmann, P. Lugli, G. Csaba: *Applications of High-Capacity Crossbar Memories in Cryptography.* IEEE Transactions on Nanotechnology 10(3), pp. 489-498, 2011.

3. H. Langhuth, S. Frederic, M. Kaniber, J. Finley, U. Rührmair: *Strong Photoluminescence Enhancement from Colloidal Quantum Dot Near Silver Nano-Island Films.* Journal of Fluorescence 21(2), pp. 539-543, 2011.

4. Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, M. Stutzmann, U. Rührmair: *Circuit-based Approaches to SIMPL Systems.* Journal of Circuits, Systems and Computers 20(1), pp. 107-123, 2011.

5. P. Lugli, A. Mahmoud, M. Algasinger, M. Stutzmann, G. Csaba, U. Rührmair: *Physical Unclonable Functions based on Crossbar Arrays for Cryptographic Applications.* International Journal of Circuit Theory and Applications 41(6), pp. 619-633, 2013.

6. U. Rührmair, M. van Dijk: *On the Practical Use of Physical Unclonable Functions in Oblivious Transfer and Bit Commitment Protocols.* Journal of Cryptographic Engineering 3(1), pp. 17-28, 2013.

7. U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, S. Devadas: *PUF Modeling Attacks on Simulated and Silicon Data.* IEEE Transactions on Information Forensics and Security 8(11), pp. 1876-1891, 2013.

Conferences:

8. 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 (ISIC), pp. 121-124, 2009.

9. S. Zmudzinski, M. Steinebach, S. Katzenbeisser, U. Rührmair: *Audio watermarking forensics: detecting malicious re-embedding.* IS&T/SPIE Electronic Imaging Conference – Media Forensics and Security XII, 2010.

10. U. Rührmair, C. Jaeger, C. Hilgers, M. Algasinger, G. Csaba, M. Stutzmann: *Security Applications of Diodes with Unique Current-Voltage Characteristics.* 14th International Conference on Financial Cryptography and Data Security (FC), 2010. Lecture Notes in Computer Science, Volume 6052, pp. 328-335, Springer, 2010.

11. 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.* 12th IEEE International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA), pp. 1-6, 2010.

12. U. Rührmair, Q. Chen, M. Stutzmann, P. Lugli, U. Schlichtmann, G. Csaba: *Towards Electrical, Integrated Implementations of SIMPL Systems.* 8th Workshop in Information Security Theory and Practice (WISTP), 2010. Lecture Notes in Computer Science, Volume 6033, pp. 277 - 292, Springer, 2010.

13. U. Rührmair, S. Katzenbeisser, M. Steinebach, S. Zmudzinski: *Watermark-Based Authentication and Key Exchange in Teleconferencing Systems.* 11th Conference on Communications and Multimedia Security (CMS), 2010. Lecture Notes in Computer Science, Volume 6109, pp. 75 - 80, Springer, 2010.

14. U. Rührmair: *Oblivious Transfer based on Physical Unclonable Functions (Extended Abstract).* 3rd International Conference on Trust and Trustworthy Computing (TRUST), 2010. Lecture Notes in Computer Science, Volume 6101, pp. 430 - 440, Springer, 2010.

15. F. Sehnke, C. Osendorfer, J. Sölter, J. Schmidhuber, U. Rührmair: *Policy Gradients for Cryptanalysis.* 20th International Conference on Artificial Neural Networks (ICANN), 2010. Lecture Notes in Computer Science, Volume 6354, pp. 168-177, Springer, 2010.

16. 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 (CCS), pp. 237-249, 2010.

17. U. Rührmair: *SIMPL Systems, Or: Can We Design Cryptographic Hardware without Secret Key Information?* 37th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM), 2011. Lecture Notes in Computer Science, Volume 6543, pp. 26-45, Springer, 2011.

18. U. Rührmair, C. Jaeger, M. Algasinger: *An Attack on PUF-based Session Key Exchange, and a Hardware-based Countermeasure: Erasable PUFs.* 15th International Conference on Financial Cryptography and Data Security (FC), 2011. Lecture Notes in Computer Science, Volume 7035, pp. 190-204, 2012.

19. Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, U. Rührmair: *The Bistable Ring PUF: A New Architecture for Strong Physical Unclonable Functions*. 4th IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 134-141, 2011.

20. Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, U. Rührmair: *Characterization of the Bistable Ring PUF.* 17th Design, Automation and Test in Europe (DATE), pp. 1459-1462, 2012.

21. U. Rührmair, M. van Dijk: *Practical Security Analysis of PUF-based Two-Player Protocols*. 14th Workshop on Cryptographic Hardware and Embedded Systems (CHES), 2012. Lecture Notes in Computer Science, Volume 7428, pp. 251-267, Springer, 2012.

22. U. Rührmair, M. van Dijk: *PUFs in Security Protocols: Attack Models and Security Evaluations.* 34th IEEE Symposium on Security and Privacy (Oakland), pp. 286-300, 2013.

23. M. van Dijk, U. Rührmair: *Protocol Attacks on Advanced PUF Protocols and Countermeasures.* 17th Design, Automation and Test in Europe (DATE), pp. 1-6, 2014.

24. U. Rührmair, D.E. Holcomb: *PUFs at a Glance.* 17th Design, Automation and Test in Europe (DATE), pp. 1-6, 2014.

25. U. Rührmair, U. Schlichtmann, W. Burleson: *Special Session: How Secure are PUFs Really? On the Reach and Limits of Recent PUF Attacks.* 17th Design, Automation and Test in Europe (DATE), pp. 1-6, 2014.

26. U. Rührmair, J. Sölter: *PUF Modeling Attacks: An Introduction and Overview*. 17th Design, Automation and Test in Europe (DATE), pp. 1-6, 2014.

27. M. van Dijk, U. Rührmair: *PUF Interfaces and Their Security*. IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2014.

28. U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, M. Majzoobi, F. Koushanfar, W. Burleson: *Efficient Power and Timing Side Channels for Physical Unclonable Functions*. CHES 2014.

29. U. Rührmair, J.L. Martinez-Hurtado, X. Xu, C. Kraeh, C. Hilgers, D. Kononchuk, J. Finley, W. Burleson: *Virtual Proofs of Reality and Their Physical Implementation*. IEEE Symposium on Security and Privacy ("Oakland"), 2015.

30. R. Horstmeyer, S. Assawaworrarit, U. Rührmair, C. Yang: *Physically secure and fully reconfigurable data storage using optical scattering*. HOST 2015

31. X. Xu, U. Rührmair, D.E. Holcomb, W.P. Burleson: *Security Evaluation and Enhancement of Bistable Ring PUFs*. RFIDSec 2015.

32. Q. Chen, U. Rührmair, S. Narayana, U. Sharif, U. Schlichtmann: *MWA Skew SRAM Based SIMPL Systems for Public-Key Physical Cryptography*. TRUST 2015.

33. S. Philippe, M. Kütt, M. McKeown, U. Rührmair, A. Glaser: *The Application of Virtual Proofs of Reality to Nuclear Safeguards and Arms Control Verifications*. 57th Annual INMM Meeting, 24-28 July 2016, Atlanta, Georgia.

Invited Book Chapters:

34. U. Rührmair, H. Busch, S. Katzenbeisser: *Strong PUFs: Models, Constructions and Security Proofs.* In: Towards Hardware Intrinsic Security. A.-R. Sadeghi, P. Tuyls (Ed.), pp. 79-96, Springer, 2010.

35. U. Rührmair, S. Devadas, F. Koushanfar: *Security based on Physical Unclonability and Disorder.* In: Introduction to Hardware Security and Trust, M. Tehranipoor and C. Wang (Ed.), pp. 65-102, Springer, 2012.

36. U. Rührmair: *SIMPL Systems as a Keyless Cryptographic and Security Primitive.* Cryptography and Security 2012. Lecture Notes in Computer Science, Vol. 6805, pp. 329-354, Springer, 2012.

37. U. Rührmair: *Disorder-based Security Hardware: An Overview.* In: Secure System Design and Trustable Computing, M. Potkonjak and C.-H. Chang (Ed.), Springer, 2016.

Preprints:

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

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

40. U. Rührmair, Q. Chen, P. Lugli, M. Stutzmann, G. Csaba: *Towards Electrical, Integrated Implementations of SIMPL Systems.* Cryptology ePrint Archive, Report 2009/278, 2009.

41. G. Csaba, X. Ju, Q. Chen, W. Porod, J. Schmidhuber, U. Schlichtmann, P. Lugli, U. Rührmair: *On-Chip Electric Waves: An Analog Circuit Approach to Physical Unclonable Functions*. IACR Cryptology ePrint Archive, Report 2009/246, 2009.

42. U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, J. Schmidhuber: *Modeling Attacks on Physical Unclonable Functions.* IACR Cryptology ePrint Archive, Report 2010/251, 2010.

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

44. U. Rührmair: *SIMPL Systems as a Keyless Cryptographic and Security Primitive.* IACR Cryptology ePrint Archive, Report 2011/189, 2011.

45. M. van Dijk, U. Rührmair: *Physical Unclonable Functions in Cryptographic Protocols: Security Proofs and Impossibility Results.* IACR Cryptology ePrint Archive, Report 2012/228, 2012.

46. U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, S. Devadas: *PUF Modeling Attacks on Simulated and Silicon Data.* IACR Cryptology ePrint Archive, Report 2013/112, 2013.

47. U. Rührmair, C. Hilgers, S. Urban, A. Weiershäuser, E. Dinter, B. Forster, C. Jirauschek: *Optical PUFs Reloaded.* IACR Cryptology ePrint Archive, Report 2013/215, 2013.

48. A. Mahmoud, U. Rührmair, M. Majzoobi, F. Koushanfar: *Combined Modeling and Side Channel Attacks on Strong PUFs.* IACR Cryptology ePrint Archive, Report 2013/632, 2013.

49. U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, F. Koushanfar, W. Burleson: *Power and Timing Side Channels for PUFs and their Efficient Exploitation.* IACR Cryptology ePrint Archive, Report 2013/851, 2013.

50. U. Rührmair: *Virtual Proofs of Reality*. IACR Cryptology ePrint Archive, Report 2014/415, 2014.

51. X. Xu, U. Rührmair, D.E. Holcomb, W.P. Burleson: *Security Evaluation and Enhancement of Bistable Ring PUFs*. Cryptology ePrint Archive, Report 2015/443, 2015.

52. C. Jin, X. Xu, W.P. Burleson, U. Rührmair, M. van Dijk: *PLayPUF: Programmable Logically Erasable PUFs for Forward and Backward Secure Key Management*. Cryptology ePrint Archive, Report 2015/1052, 2015.

53. U. Rührmair: *On the Security of PUF Protocols under Bad PUFs and PUFs-inside-PUFs Attacks*. Cryptology ePrint Archive, Report 2016/322, 2016.

According to Google scholar, the above publications have been quoted around 1300 times altogether (status: September 26, 2016) [26].

Bibliography

We stress again that this is a cumulative thesis. The bibliography below therefore only contains the references used in Chapter 1 and all publications of the candidate. Any other references are directly given at the end of the respective chapters.

[1] F. Armknecht, R. Maes, A.-R. Sadeghi, B. Sunar, P. Tuyls: *Memory Leakage-Resilient Encryption Based on Physically Unclonable Functions*. ASIACRYPT 2009, pp. 685-702, 2009.

[2] F. Armknecht, R. Maes, Ahmad-Reza Sadeghi, F.-X. Standaert, C. Wachsmann: *A Formal Foundation for the Security Features of Physical Functions*. IEEE Symposium on Security and Privacy 2011, pp. 397-412, 2011.

[3] Y. Aumann, Y. Z. Ding, M. O. Rabin: *Everlasting security in the bounded storage model*. IEEE Transactions on Information Theory, Vol. 48(6), pp. 1668-1680, 2002.

[4] D. J. Bernstein, J. Buchmann, E. Dahmen (Ed.): *Post-Quantum Cryptography*. Springer, 2009. ISBN 978-3-540-88701-0.

[5] C.H. Bennett, G. Brassard: *Quantum cryptography: Public key distribution and coin tossing.* IEEE International Conference on Computers, Systems and Signal Processing, Vol. 175(150), p. 8, 1984.

[6] C. Bruzska, M. Fischlin, H. Schröder, S. Katzenbeisser: *Physical Unclonable Functions in the Universal Composition Framework.* CRYPTO 2011, pp. 51-70, 2011.

[7] 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 (ISIC), pp. 121-124, 2009.

[8] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, U. Rührmair: *The Bistable Ring PUF: A New Architecture for Strong Physical Unclonable Functions*. 4th IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 134-141, 2011.

[9] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, U. Rührmair: *Characterization of the Bistable Ring PUF.* 17th Design, Automation and Test in Europe (DATE), pp. 1459-1462, 2012.

[10] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, M. Stutzmann, U. Rührmair: *Circuit-based Approaches to SIMPL Systems.* Journal of Circuits, Systems and Computers 20(1), pp. 107-123, 2011.

[11] W.E. Cobb, E.D. Laspe, R.O. Baldwin, M.A. Temple, Y.C. Kim: *Intrinsic Physical-Layer Authentication of Integrated Circuits*. IEEE Transactions on Information Forensics and Security, Vol. 7(1), pp. 14-24, 2012.

[12] C. Crepeau: *Efficient cryptographic protocols based on noisy channels*. EUROCRYPT 1997, pp. 306-317, 1997.

[13] C. Crepeau, K. Morozov, S. Wolf: *Efficient Unconditional Oblivious Transfer from Almost Any Noisy Channel*. SCN 2004, pp. 47-59, 2004.

[14] G. Csaba, X. Ju, Q. Chen, W. Porod, J. Schmidhuber, U. Schlichtmann, P. Lugli, U. Rührmair: *On-Chip Electric Waves: An Analog Circuit Approach to Physical Unclonable Functions*. IACR Cryptology ePrint Archive, Report 2009/246, 2009.

[15] 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.* 12th IEEE International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA), pp. 1-6, 2010.

[16] D. Dachman-Soled, N. Fleischhacker, J. Katz, A. Lysyanskaya, D. Schröder: *Feasibility and Infeasibility of Secure Computation with Malicious PUFs*. CRYPTO 2014, pp. 405-420, 2014.

[17] I. Damgard, A. Scafuro: *Unconditionally Secure and Universally Composable Commitments from Physical Assumptions*. ASIACRYPT 2013, pp. 100-119, 2013.

[18] M. van Dijk, U. Rührmair: *Physical Unclonable Functions in Cryptographic Protocols: Security Proofs and Impossibility Results*. IACR Cryptology ePrint Archive, Report 2012/228, 2012.

[19] M. van Dijk, U. Rührmair: *Protocol Attacks on Advanced PUF Protocols and Countermeasures.* 17th Design, Automation and Test in Europe (DATE), pp. 1-6, 2014.

[20] M. van Dijk, U. Rührmair: *PUF Interfaces and Their Security*. IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2014 (to appear).

[21] Y. Z. Ding, D. Harnik, A. Rosen, R. Shaltiel: *Constant-Round Oblivious Transfer in the Bounded Storage Model*. J. of Cryptology 20(2), pp. 165-202, 2007.

[22] B. Gassend, D.E. Clarke, M. van Dijk, S. Devadas: *Silicon physical random functions*. ACM CCS 2002, pp. 148-160, 2002.

[23] B. Gassend, M. van Dijk, D.E. Clarke, E. Torlak, S. Devadas, P. Tuyls: *Controlled physical random functions and applications*. ACM Transactions on Information and System Security, Vol. 10(4), 2008.

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

[25] O. Goldreich: *Foundations of Cryptography - A Primer*. Foundations and Trends in Theoretical Computer Science 1(1), Now Publishers, 2005.

[26] See the Google scholar profile of the candidate (status: September 26, 2015): https://scholar.google.de/citations?user=5T2i3XoAAAAJ&hl=de&oi=ao

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

[28] C. Jaeger, M. Algasinger, U. Rührmair, G. Csaba, M. Stutzmann: *Random p-n-junctions for physical cryptography.* Applied Physics Letters 96, 172103, 2010.

[29] J. Kilian: *Founding cryptography on oblivious transfer*. STOC 1988, pp. 20-31, 1988.

[30] H. Langhuth, S. Frederic, M. Kaniber, J. Finley, U. Rührmair: *Strong Photoluminescence Enhancement from Colloidal Quantum Dot Near Silver Nano-Island Films.* Journal of Fluorescence 21(2), pp. 539-543, 2011.

[31] K.P. Liessmann: *Einsamkeit und Freiheit. Zur Renaissance akademischer Bildung*. Lecture at the "Carl Friedrich von Siemens Stiftung", Schloss Nymphenburg, München, May 12, 2014.

[32] H.-K. Lo, H. F. Chau: *Why quantum bit commitment and ideal quantum coin tossing are impossible.* Physica D: Nonlinear Phenomena 120.1, pp. 177-187, 1998.

[33] P. Lugli, A. Mahmoud, M. Algasinger, M. Stutzmann, G. Csaba, U. Rührmair: *Physical Unclonable Functions based on Crossbar Arrays for Cryptographic Applications.* International Journal of Circuit Theory and Applications 41(6), pp. 619-633, 2013.

[34] D. Mayers: *Unconditionally secure quantum bit commitment is impossible.* Physical review letters 78.17: 3414, 1997.

[35] A. Mahmoud, U. Rührmair, M. Majzoobi, F. Koushanfar: *Combined Modeling and Side Channel Attacks on Strong PUFs.* IACR Cryptology ePrint Archive, Report 2013/632, 2013.

[36] A. Maiti, I. Kim, P. Schaumont: *A Robust Physical Unclonable Function With Enhanced Challenge-Response Set*. IEEE Transactions on Information Forensics and Security, Vol 7(1), pp. 333-345, 2012.

[37] A. Maiti, P. Schaumont: *Improved Ring Oscillator PUF: An FPGA-friendly Secure Primitive*. Journal of Cryptology, Vol. 24(2), pp. 375-397, 2011.

[38] U. Maurer: *Conditionally-perfect secrecy and a provably-secure randomized cipher*. Journal of Cryptology, Vol. 5(1), pp. 53-66, 1992.

[39] R. Ostrovsky, A. Scafuro, I. Visconti, A. Wadia: *Universally Composable Secure Computation with (Malicious) Physically Uncloneable Functions*. IACR Cryptology ePrint Archive, Report 2012/143, 2012.

[40] R. Ostrovsky, A. Scafuro, I. Visconti, A. Wadia: *Universally Composable Secure Computation with (Malicious) Physically Uncloneable Functions*. EUROCRYPT 2013, pp. 702-718, 2013.

[41] R. Pappu: *Physical One-Way Functions.* PhD Thesis, MIT, 2001.

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

[43] David A. Patterson: *20th century vs. 21st century C&C: the SPUR manifesto*. Communications of the ACM 48(3), pp. 15-16, 2005.

[44] K. Rosenfeld, E. Gavas, R. Karri: *Sensor Physical Unclonable Functions*. HOST 2010, pp. 112-117, 2015.

[45] U. Rührmair: *On the Formal Foundations of Physical Unclonable Functions.* Security Hardware in Theory and Practice - A Marriage of Convenience. Event 08253, Schloss Dagstuhl, Germany, 2008.

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

[47] U. Rührmair: *Oblivious Transfer based on Physical Unclonable Functions (Extended Abstract).* TRUST 2010, pp. 430 - 440, 2010.

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

[49] U. Rührmair: *SIMPL Systems as a Keyless Cryptographic and Security Primitive.* IACR Cryptology ePrint Archive, Report 2011/189, 2011.

[50] U. Rührmair: *SIMPL Systems, Or: Can We Design Cryptographic Hardware without Secret Key Information?* 37th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM), 2011. Lecture Notes in Computer Science, Volume 6543, pp. 26-45, Springer, 2011.

[51] U. Rührmair: *SIMPL Systems as a Keyless Cryptographic and Security Primitive.* In: Cryptography and Security: From Theory to Applications — Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday, D. Naccache (Ed.), Lecture Notes in Computer Science, Vol. 6805, pp. 329-354, Springer, 2012.

[52] U. Rührmair: *Disorder-based Security Hardware.* PhD Thesis, Technical University of Munich, 2014.

[53] U. Rührmair: *Disorder-based Security Hardware: An Overview.* In: Secure System Design and Trustable Computing, M. Potkonjak and C.-H. Chang (Ed.), Springer, 2014/15, to appear.

[54] U. Rührmair: *Virtual Proofs of Reality*. IACR Cryptology ePrint Archive, Report 2014/415, 2014.

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

[56] U. Rührmair, Q. Chen, P. Lugli, M. Stutzmann, G. Csaba: *Towards Electrical, Integrated Implementations of SIMPL Systems*. Cryptology ePrint Archive, Report 2009/278, 2009.

[57] U. Rührmair, Q. Chen, M. Stutzmann, P. Lugli, U. Schlichtmann, G. Csaba: *Towards Electrical, Integrated Implementations of SIMPL Systems.* 8th Workshop in Information Security Theory and Practice (WISTP), 2010. Lecture Notes in Computer Science, Volume 6033, pp. 277 - 292, Springer, 2010.

[58] U. Rührmair, S. Devadas, F. Koushanfar: *Security based on Physical Unclonability and Disorder.* In: Introduction to Hardware Security and Trust, M. Tehranipoor, C. Wang, pp. 65-102. Springer New York, 2012.

[59] U. Rührmair, M. van Dijk: *Practical Security Analysis of PUF-Based Two-Player Protocols*. CHES 20120, pp. 251-267, CHES 2012.

[60] U. Rührmair, M. van Dijk: *PUFs in Security Protocols: Attack Models and Security Evaluations*. IEEE Symposium on Security and Privacy 2013, pp. 286-300, 2013.

[61] U. Rührmair, M. van Dijk: *On the Practical Use of Physical Unclonable Functions in Oblivious Transfer and Bit Commitment Protocols.* Journal of Cryptographic Engineering 3(1), pp. 17-28, 2013.

[62] U. Rührmair, C. Hilgers, S. Urban, A. Weiershäuser, E. Dinter, B. Forster, C. Jirauschek: *Optical PUFs Reloaded.* IACR Cryptology ePrint Archive, Report 2013/215, 2013.

[63] U. Rührmair, D.E. Holcomb: *PUFs at a Glance.* 17th Design, Automation and Test in Europe (DATE), pp. 1-6, 2014.

[64] U. Rührmair, C. Jaeger, M. Algasinger: *An Attack on PUF-Based Session Key Exchange and a Hardware-Based Countermeasure: Erasable PUFs*. Financial Cryptography 2011, pp. 190-204, 2012.

[65] U. Rührmair, C. Jaeger, M. Bator, M. Stutzmann, P. Lugli, G. Csaba: *Applications of High-Capacity Crossbar Memories in Cryptography.* IEEE Transactions on Nanotechnology 10(3), pp. 489-498, 2011.

[66] U. Rührmair, C. Jaeger, C. Hilgers, M. Algasinger, G. Csaba, M. Stutzmann: *Security Applications of Diodes with Unique Current-Voltage Characteristics*. Financial Cryptography and Data Security (FC 2010), Lecture Notes in Computer Science, Vol. 6052, pp. 328-335, Springer Verlag, 2010.

[67] U. Rührmair, S. Katzenbeisser, M. Steinebach, S. Zmudzinski: *Watermark-Based Authentication and Key Exchange in Teleconferencing Systems.* 11th Conference on Communications and Multimedia Security (CMS), 2010. Lecture Notes in Computer Science, Volume 6109, pp. 75 - 80, Springer, 2010.

[68] U. Rührmair, J.L. Martinez-Hurtado, X. Xu, C. Kraeh, C. Hilgers, D. Kononchuk, J.J. Finley, W.P. Burleson:

[69] . IEEE Symposium on Security and Privacy 2015, pp. 70-85, 2015.

[70] U. Rührmair, U. Schlichtmann, W. Burleson: *Special Session: How Secure are PUFs Really? On the Reach and Limits of Recent PUF Attacks.* 17th Design, Automation and Test in Europe (DATE), pp. 1-6, 2014.

[71] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, J. Schmidhuber: *Modeling Attacks on Physical Unclonable Functions.* IACR Cryptology ePrint Archive, Report 2010/251, 2010.

[72] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, J. Schmidhuber: *Modeling Attacks on Physical Unclonable Functions*. ACM CCS, pp. 237-249, 2010.

[73] U. Rührmair, J. Sölter: *PUF Modeling Attacks: An Introduction and Overview*. 17th Design, Automation and Test in Europe (DATE), pp. 1-6, 2014.

[74] U. Rührmair, J. Sölter, F. Sehnke: *On the Foundations of Physical Unclonable Functions.* Cryptology e-Print Archive, Report 2009/277, June 2009.

[75] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, S. Devadas: *PUF Modeling Attacks on Simulated and Silicon Data.* IACR Cryptology ePrint Archive, Report 2013/112, 2013.

[76] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, S. Devadas: *PUF Modeling Attacks on Simulated and Silicon Data*. IEEE Transactions on Information Forensics and Security, Vol. 8(11), pp. 1876-1891, 2013.

[77] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, F. Koushanfar, W. Burleson: *Power and Timing Side Channels for PUFs and their Efficient Exploitation.* IACR Cryptology ePrint Archive, Report 2013/851, 2013.

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

[79] See http://www.chesworkshop.org/former.php

[80] See http://www.hostsymposium.org/

[81] See http://www.informatik.uni-trier.de/ ley/db/conf/date/date2014.html

[82] F. Sehnke, C. Osendorfer, J. Sölter, J. Schmidhuber, U. Rührmair: *Policy Gradients for Cryptanalysis.* 20th International Conference on Artificial Neural Networks (ICANN), 2010. Lecture Notes in Computer Science, Volume 6354, pp. 168-177, Springer, 2010.

[83] A. Sharma, L. Subramanian, E.A. Brewer: *PaperSpeckle: microscopic fingerprinting of paper.* ACM CCS 2011, pp. 99-110, 2011.

[84] M. Steinebach, S. Zmudsinski, S. Katzenbeisser, U. Rührmair: *Audio watermarking forensics: detecting malicious re-embedding.* IS&T/SPIE Electronic Imaging Conference – Media Forensics and Security XII, 2010.

[85] G. E. Suh, S. Devadas: *Physical Unclonable Functions for Device Authentication and Secret Key Generation.* DAC 2007: 9-14

[86] P. Tuyls, B. Skoric: *Strong Authentication with Physical Unclonable Functions.* In: Security, Privacy and Trust in Modern Data Management, M. Petkovic, W. Jonker (Eds.), pp. 133-148, Springer, 2007.