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 Introduction and Summary in April 2011 and April 2014)

Contents

Abstract

In this paper, we introduce two formal means by which *physical* adversarial actions and features can be modeled in cryptography and security: The concepts of a *"physical Turing machine (PhTM or φ-TM)"* and of a *"technology"* on which the PhTM operates. We show by two examples how these concepts can be applied: Firstly, we sketch their use in formalizing *physical* adversarial computations (quantum computation [4], optical techniques [26, 15], etc.) in classical cryptpography, which an adversary might carry out to attack complexity-based schemes. Secondly, we work out in more detail the application of PhTMs in the formal treatment of physical unclonable functions and physical cryptography in general, in which disordered, unclonable physical objects are used for cryptographic purposes. PhTMs allow a rigid formal expression of the required properties of these objects (such as their physical unclonability), and enable us to lead formal reductionist proofs in this field. The hybrid nature of PhTMs thereby allows us to combine physical with computational assumptions in the proof. As an example, we lead a formal proof of a physical scheme that combines a classical digital signature with an unclonable, unique object in order to "label" or "tag" valuable objects securely and in a forgery-proof manner.

We stress that PhTMs as introduced in this paper cannot directly and straightforwardly answer the question which physical tasks are eventually feasible and infeasible in our universe. But such an expectation would be unreasonably high; recall that classical Turing machines also do not allow to draw a simple line between feasible and infeasible computations, as the NP vs. P issue shows. Rather, they provide us with a formal backbone in which relevant physical security features can be expressed and security proofs can be led. Apart from the applications sketched in this paper, many other uses of PhTMs lie at hand, for example in defining security against side channels or invasive attacks, or in the development of a *"physical"* structural complexity theory, which are left to future work.

1 Introduction and Overview

Physical Cryptography (PhC). One recent trend in cryptography and security is to utilize "disordered", randomly structured physical systems for cryptographic and security purposes. Some of the best-known concepts within the area include physical unclonable functions (PUFs) [19, 12], certificates of authenticity (COAs) [10], or public PUFs [3]. In lack of an established name, the subfield comprising of all of these physical, disordered primitives has at times been termed *"physical cryptography (PhC)"* [23], or also "*disorder-based security*" [23]. Central to this young field is the idea that not only the mathematical properties of certain functions (such as non-invertability, pseudo-randomness, etc.) have cryptographic applicability. Instead, also the properties of disordered, randomly structured physical systems can be well exploited for cryptographic and security purposes [19, 12]. One example is their physical unclonability, i.e., the fact that the uncontrollable, small-scale manufacturing variations that inevitably occur in the production of each physical object cannot be cloned or reproduced with perfect accuracy, not even by a well-equipped adversary, and that they make each object individual and unique [10, 12], even nominally identically designed objects. Another example is the high information content or entropy that can occur in randomly disorder physical media. It can be generated easily and inexpensively, but which cannot be read-out or characterized completely in a feasible time period by a fraudster [19].

There are two potential advantages that result from this novel approach. First of all, it can enable a better protection of secret keys in cryptographic hardware. Instead of storing secret keys in vulnerable non-volatile digital memory, the keys are derived from, or hidden in, the analog characteristics of a randomly structured medium. This makes them harder to read out, derive, or obtain otherwise for the adversary (see again [23] for an overview and detailed discussion).

Secondly, the utilization of physical disorder and physical cryptography can sometimes avoid the classical, unproven computational assumptions like the purported hardness of the factoring and discrete logarithm problem. It trades these assumptions against hypotheses about the employed, disordered physical systems, for example their unclonability, their input-output complexity, or the numeric unpredictability of their output. PhC thereby creates an alternative foundation for cryptography and security, which rests on assumptions that are independent from the classical, computational assumptions of standard cryptography. In this sense, it is in alignment to other, non-standard approaches, such as quantum cryptography [4], post-quantum crypto [7], noise-based cryptography [17], or the bounded storage model [16]. In fact, it has been shown by several authors how classical cryptographic protocols can be based on PUFs only, without involving further computational assumptions. This includes PUF-based schemes for oblivious transfer, key exchange, or bit commitment [21, 5, 18].

The abovementioned advantages have led to an strongly increasing interest in the novel subfield of PUFs and physical cryptography over recent years. Naturally, also the field's formalization has attracted considerable attention, partly leading to publications at major conferences like Crypto, Eurocrypt, or IEEE S&P [22, 2, 5, 18, 24]. However, none of the existing approaches has developed a *formal* machine model that could

- model the physical attack capabilities of an adversary,
- model the physical unclonability of the used disordered primitives like PUFs,
- and, at the same time, be employed in formal security proofs, for example reductionist approaches.

For example, the information-theoretic formulation of PUF-security that is given by Brzuska et al. [5], and which has been continued by Ostrovsky et al. [18], 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. Furthermore, the formal framework of Armknecht et al. [2] does not stipulate a machine model upon which formal security proofs could be based, and for this reason their definitions indeed have not been used in any formal security proofs up to this date. We try to address this issue in this paper.

Our Contributions. We make three contributions in this work. First of all, we describe an extension of the Turing machine model which we call *Physical Turing Machines* (or *φ-TMs* 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. They allow us to model cryptographic parties that use physical mechanisms in one way or the other in their protocols or in their attacks. We argue that this new formal model could potentially be used for at least three purposes:

(i) To formally model the security of classical, mathematical cryptoschemes against computational attacks that employ *physical* computations, such as quantum computers or optical computing devices, and to lead reductionist security proofs in this model.

(ii) To formally model the security of cryptographic hardware against physical attacks on all levels, including invasive, side channel and fault injection attacks, and to lead reductionist security proofs in this model.

(iii) To formalize physical cryptography, and to lead comprehensive reductionist security proofs in this area. In particular, PhTMs allow to model the *physical* unclonability of the employed physical objects such as PUFs, in opposition to existing approaches (see above).

Secondly, we briefly discuss how Physical Turing Machines can be applied to the above purpose (i), and sketch how the foundations of classical cryptography could be reformulated by use of Physical Turing Machines. This reformulation allows us to include attacks by quantum computers, for example. One particular benefit of using Physical Turing Machines in this reformulation is that they allow us to take the limited state of current technology into account. Simply substituting standard Turing Machines by Quantum Turing Machines would overestimate the power of the attacker, and would leave many currently existing schemes unrightfully insecure. Simply using standard Turing machinesm, on the other hand, excludes any attacks that employ other computing models. Using PhTMs relative to some given state of technology takes a medium position here, as we argue in all detail later.

Thirdly, we exemplarily apply PhTMs to the above item (iii), and we lead a detailed security proof for a well-known scheme in physical cryptography. This 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. 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 said method 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 Physical Turing Machines as the underlying model.

Two benefits of using PhTMs in this context are that they (i) enable us to formally state the physical non-clonability of physical objects; and (ii) that they allow us to reconcile the inherently finite nature of a physical object with the inherently asymptotic formalization of digital signature schemes. These features allow us to lead the aspired formal security proof.

We would like conclude this paragraph by delimiting too excessive expectations of the readers already at this early point. Our aim in this paper is not to assess the computational power of physical systems in a strict sense, or to prove unconditional feasibility/infeasibility results about the (computational) power of arbitrary physical systems or devices. Given the current state of the complexity theory and the still unresolved P vs. NP question, such a hope would indeed be unreasonably high. Our main focus rather is to provide a formal foundations and backbone for reductionist security proofs in physical cryptography and related disciplines. As it turns out, this goal is complicated 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.) on physical objects. In Section 4, we briefly discuss the application of PhTMs to the formalization of classical 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 [27]. First of all, both computability theory and complexity theory often implicitly assume the validity of the well-known Church-Turing Thesis (CT) [27]. Secondly, complexity theory in large parts also assumes the (perhaps less well known) Extended Church-Turing Thesis (ECT), which can be formulated like this [27]:

> **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) [19, 12] and certificates of authenticity (COAs) [10], but there are also others; a good overview is given in [23]. As mentioned in the introduction, the corresponding research area could be (and has been) termed *"physical cryptography"* or *"disorder-based security"*.

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" [23] 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 [23, 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 [23]). 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 world-wide 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 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.

[1]Please note that this is in opposition to the approach of optical PUFs of Pappu et al. [19], 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.

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 [19, 12, 23]).

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 [19], 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 [23, 22, 25].

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 [23].

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 [21, 5, 18]. 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"*. 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 $\mathcal{U} = (\mathcal{O}, \mathcal{D}, \mathcal{B}, m)$ 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 $X = (s, O_1, \ldots, O_n)$ instead of $X = (s, (O_1, \ldots, O_n))$.*
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 $\mathcal{U} = (\mathcal{O}, \mathcal{D}, \mathcal{B}, m)$ 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 $O_1, O_2, \ldots$ 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 $(O_1, \ldots, O_k)$.
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 φ-TM are a mixture of numbers and physical objects, we should define a norm on such inputs.

Definition 3.4 (Input and Output Norm)**.** *Let $\mathcal{U} = (\mathcal{O}, \mathcal{D}, \mathcal{B}, m)$ be a universe. We define a norm $\|\cdot\|$ on all elements $X = (x, (O_1, \ldots, O_n))$ 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 $\mathcal{U} = (\mathcal{O}, \mathcal{D}, \mathcal{B}, m)$ 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 φ-TM has got a counter, which cannot be accessed or deliberately altered by the φ-TM 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 φ-TM 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 φ-TM 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 φ-TM.

Notation 3.6 (Output of physical Turing machines)**.** *Let M be a φ-TM 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 φ-TM to be ressource efficient; as usual, ressource efficiency is taken as being polynomially bounded in some way.

Definition 3.7 (Polynomial φ-TMs)**.** *Let $\mathcal{U}$ be a universe, and let $M = (P, M_1, \ldots, M_n)$ be a φ-TM 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 φ-TM 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 φ-TM if is both a polynomial time and a polynomial mass φ-TM.*

Finally, a computation executed by a φ-TM M is called a polynomial time or polynomial mass or polynomial computation if M is a polynomial time or polynomial mass or polynomial φ-TM, respectively.

Now we can state an important feature of φ-TMs, 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 $\mathcal{U} = (\mathcal{O}, \mathcal{D}, \mathcal{B}, m)$ 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 φ-TM respects a state of technology.

Definition 3.9 (φ-TMs 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 φ-TM in $\mathcal{U}$. We say that M respects $\mathcal{T}$, or that M is a φ-TM 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 φ-TMs 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 φ-TMs 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 φ-TMs, 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 φ-TM 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 φ-TM in order to restrict

(A) the set of possible objects and possible machines in the universe $\mathcal{U}$, from which a φ-TM might draw its physical parts.

(B) the technology $\mathcal{T}$ from which a φ-TM 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 φ-TMs, or of polynomial φ-TMs, 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 φ-TMs, then this theory might be subject to change or even replacement over time. The general formalism provided by φ-TMs, however, can persist independent of such change: We simply determine the universes $\mathcal{U}$ in which the φ-TMs 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: φ-TM 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 φ-TM. 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 φ-TMs 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, probabilistic physical Turing machines will be more important for that purpose than (standard) physical Turing machines. Oracle physical Turing machines are probabilistic φ-TMs 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 probabilistic physical 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 φ-TMs 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 $\mathcal{U} = (\mathcal{O}, \mathcal{D}, \mathcal{B}, m)$ 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:*

$$\mathcal{F} : \mathcal{M} \longrightarrow \left\{ (\Omega, \mathcal{A}, P_M) \;\middle|\; \begin{array}{l} \Omega \;=\; \{0,1\}^* \times \mathcal{O}^* \times \{0,1\}^* \times \mathcal{O}^* \times \mathbb{N} \\ \mathcal{A} \;=\; \mathcal{A}\big(\{F \mid F \textit{ is a finite subset of } \Omega\}\big) \\ (\Omega, \mathcal{A}, P_M) \textit{ is a probability space} \end{array} \right\}$$

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 probabilistic physical oracle Turing machines, or φ-POTMs.

Definition 3.16 (Probabilistic Physical Oracle Turing Machines)**.** *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a function. An probabilistic physical oracle Turing machine or φ-POTM M with an oracle for f is a φ-PTM 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 φ-PTM is basically as the same as for standard TMs. The familiar terms 'oracle call' and 'number of oracle calls' can be used for φ-POTMs in the straightforward way.

3.5 Object Generators and Measuring Devices

We will introduce two special types of φ-TMs 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 φ-TM 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 φ-TM 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 φ-TM 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 $p_1, \ldots, p_n$ be measurement parameters of M. For notational ease, we introduce the following abbreviations:*

$$\begin{aligned}
M(p, O) &\overset{\text{def}}{=} \Pi^1(Out_M(p, O)) && (*)\\
\overline{p} &\overset{\text{def}}{=} (p_1, \ldots, p_n)\\
\overline{p}_i &\overset{\text{def}}{=} (p_{(i,1)}, \ldots, p_{(i,n_i)})\\
M(\overline{p}, O) &\overset{\text{def}}{=} (M(p_1, O), \ldots, M(p_n, O))\\
M(\overline{p}_i, O) &\overset{\text{def}}{=} (M(p_{(i,1)}, O), \ldots, M(p_{(i,n_i)}, O))\\
\overline{P}(O) &\overset{\text{def}}{=} (\overline{p}, M(\overline{p}, O)\\
\overline{P}_i(O) &\overset{\text{def}}{=} (\overline{p}_i, M(\overline{p}_i, O)\\
&= (p_{(i,1)}, \ldots, p_{(i,n_i)}, M(p_{(i,1)}, O), \ldots, M(p_{(i,n_i)}, O))
\end{aligned}$$

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 could, in principle, be attacked by quantum algorithms; what saves their security is the current state of technology, not a formal mathematical or Turing-machine based argument.

We will start by repeating some facts from binary cryptography.

Definition 4.1 (Signature Schemes)**.** *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.

The definition does not tell us anything about the security of digital signatures. The attack scenario commonly associated with digital signatures is that of an adaptive chosen message attack. This is a certain attack scenario the adversary is placed in, together with a definition what a successful attack of the adversary in that scenario is. A digital signature scheme which does not allow successful attacks in the scenario is consequently regarded as secure.

The adaptive chosen message attack scenario can be described as follows: Given a signature scheme (G, Sig, Ver), the key generator is run with input 1^n to produce a private key s and a public key v. Then, the attacker is allowed to ask for polynomially many signatures $Sig(s, \alpha_i)$ for polynomially many binary strings α_i, $i = 1, \dots, p(n)$, which he may choose adaptively. After he has received all the asked signatures, he must try to output a valid signature for a new string $\alpha \neq \alpha_i$, $i = 1, \dots, p(n)$. If he can do so, that is, if he can output a pair (α, S) such that $Ver(v, \alpha, S) = 1$ and $\alpha \neq \alpha_i$ for all $i = 1, \dots, p(n)$, then he is said to have broken the signature scheme.

The following standard definition models this scenario by using oracle Turing machines. The attacker is modeled as a polynomial time oracle Turing machine with an oracle for the function $Sig(s, \cdot)$. Thereby the following notation is used: S_s denotes the (random) oracle for the function $Sig(s, \cdot)$, and $Q_M^{S_s}$ denotes the set of queries made by machine M to the oracle S_s.

Definition 4.2 (Secure Signature Schemes). *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*

$$\Pr\left[\begin{array}{c} Ver(v,\alpha,\beta) = 1 \text{ and } \alpha \notin Q_M^{S_s}(v), \\ \text{where } (s,v) \leftarrow G(1^n) \text{ and } (\alpha,\beta) \leftarrow M^{S_s}(v) \end{array}\right] < p(n),$$

where the probability is taken over the coin toses of algorithms G, M and Veras 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)). *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,

$$\Pr\left[\begin{array}{c|c} M(n) \text{ outputs a nontrivial} & n \text{ is sampled uniformly at} \\ \text{divisor of } n & \text{random from the set } H_k \end{array}\right] < \nu(k).$$

Now we can state the following theorem by [GoldwasserMicaliRivest].

Theorem 4.4 (Secure Signatures from the IAF). *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 $\mathcal{U} = (\mathcal{O}, \mathcal{D}, \mathcal{B}, m)$ 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 φ-TMs.

Definition 4.6 (φ-Intractability Assumption for Factoring (φ-IAF))**.** *Let $\mathcal{U} = (\mathcal{O}, \mathcal{D}, \mathcal{B}, m)$ 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 $\mathcal{U} = (\mathcal{O}, \mathcal{D}, \mathcal{B}, m)$ 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 φ-TMs 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 φ-TMs to standard cryptography. We will proceed with the application of φ-TMs to physical cryptography, where the property of φ-TMs 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 llike, 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 paramters 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*

$$CLONE : \left\{ (1^k, \overline{P}(O), O) \;\middle|\; O \in \mathcal{O} \text{ and } \overline{p} \in (D_M^P)^* \right\} \longrightarrow \mathcal{O}^2,$$

for any polynomial p and for any sufficiently large k,

$$\Pr\left[\begin{array}{l} \overline{P}(O) = \overline{P}(O_1) = \overline{P}(O_2) \\ \qquad \textit{where } (O_1, O_2) \leftarrow \mathit{CLONE}(\overline{P}(O), O) \\ \qquad \textit{and } (\overline{P}(O), O) \leftarrow \mathit{OG}(1^k) \end{array}\right] \leq 1/p(k)$$

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 φ-TMs 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 φ-TM CLONE, so that they can only have some suppesedly 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 $\mathit{OG}(1^k)$ 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 $OS = (\mathit{OG}, M)$ be a unique object system in $\mathcal{T}$. Let further p and q be polynomials, and let $\mathit{OG}_1, \ldots, \mathit{OG}_{p(k)}$ be object generators in $\mathcal{T}$. Then, the following holds:*

1. *For all polynomials r and for all sufficiently large k,*

$$\Pr\left[\begin{array}{l} \exists\, i \neq 1 :\ \overline{P}_1(O_1) = \overline{P}_i(O_i) \\ \qquad \textit{where } \big(\overline{P}_i(O_i), O_i\big) \leftarrow \mathit{OG}(1^k) \textit{ for } i = 1, \ldots, q(k) \end{array}\right] \leq 1/r(k).$$

2. *For all polynomials r and for all sufficiently large k,*

$$\Pr\left[\begin{array}{l} \exists\, i, j, l :\ \overline{P}_i(O_i) = \overline{P}_{(j,l)}(O_{(j,l)}) \\ \qquad \textit{where } \big(\overline{P}_{(j,l)}(O_{(j,l)}), O_{(j,l)}\big) \leftarrow \mathit{OG}_j(1^k) \\ \qquad \textit{for } j = 1, \ldots, p(k) \textit{ and } l = 1, \ldots, q(k). \end{array}\right] \leq 1/r(k).$$

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

$$\Pr\left[\begin{array}{l} \exists\, i \neq 1 :\ \overline{P}_1(O_1) = \overline{P}_i(O_i) \\ \qquad \text{where } \big(\overline{P}_i(O_i), O_i\big) \leftarrow \mathit{OG}(1^k) \text{ for } i = 1, \ldots, q(k) \end{array}\right] > 1/r(k).$$

Then, we construct a φ-PTM *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,

$$\Pr\left[\begin{array}{l} \overline{P}(O) = \overline{P}(O_1) = \overline{P}(O_2) \\ \qquad \text{where } (O_1, O_2) \leftarrow CLONE(\overline{P}(O), O) \\ \qquad \text{and } (\overline{P}(O), O) \leftarrow OG(1^k) \end{array}\right] \leq \nu(n)$$

where the probability is taken over the random outputs of *CLONE* and *OG*.

□

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 φ-TM $TWIN$ in $\mathcal{T}$ with*

$$TWIN : \left\{ 1^k \,\middle|\, k \in \mathbb{N} \right\} \longrightarrow \left\{ \left(\overline{P}(O_1), O_1, \overline{P}(O_2), O_2\right) \;\middle|\; O_1, O_2 \in \mathcal{O},\ \overline{p} \in (D_M^P)^* \right\},$$

   for any polynomial p and for all sufficiently large k, it holds that

$$\Pr\left[\begin{array}{c} \overline{P}(O_1) = \overline{P}(O_2) \text{ and} \\ \left(\overline{P}(O_1), O_1\right), \left(\overline{P}(O_2), O_2\right) \in \mathsf{Range}(OG) \setminus \{(\lambda, \lambda_O)\} \\ \text{where } \left(\overline{P}(O_1), O_1, \overline{P}(O_2), O_2\right) \leftarrow TWIN(1^k) \end{array}\right] \leq 1/p(k)$$

   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*

$$\Pi^1(\mathsf{Range}(OG)) \setminus \left\{(\lambda, \lambda_O)\right\}.$$

   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

$$(x, O) \in \mathsf{Range}(OG) \setminus \left\{(\lambda, \lambda_O)\right\}.$$

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 resistancy 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 manufacturer resistancy 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 resistancy as in Definition 5.4 answers an issue left open in [12, 11], where the property of manufacturer resistancy 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 resistancy 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 resistancy 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" [24].

We suggest therefore contrary to [12], that the notion of manufacturer resistancy should in general only be used in connection with a corresponding verifiability condition, and did not define manufacturer resistancy or manufacturer resistant object systems, as independent notions of their own.

There is another issue about the manufacturer resistancy 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 φ-PTM *CLONE* such that for infinitely many k and a fixed polynomial p^* it holds that

$$\Pr\left[\begin{array}{l}\overline{P}(O)=\overline{P}(O_1)=\overline{P}(O_2)\\ \qquad \text{where } (O_1,O_2)\leftarrow \mathit{CLONE}(\overline{P}(O),O)\\ \qquad \text{and } (\overline{P}(O),O)\leftarrow OG(1^k)\end{array}\right] > 1/p^*(k),$$

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 φ-PTM *TWIN* which violates the property that (OG,M) is a unique object system. We define *TWIN* as follows:

MACHINE *TWIN*:

```
Input  $1^k$
    Set  $(\overline{P}(O),O)\leftarrow OG(1^k)$
    Set  $(O_1,O_2)\leftarrow \mathit{CLONE}(1^k,\overline{P}(O),O)$
Output  $(\overline{P}(O_1),O_1,\overline{P}(O_2),O_2)$
```

It is obvious from the definition of *TWIN* that if *CLONE*, OGand M are polynomial φ-PTM 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 resistancy 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 resistancy 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 φ-TM 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 resistancy 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 φ-TMs 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 φ-TMs 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 φ-TM FAKE in $\mathcal{T}$ and for any polynomial p there is a negligible function ν such that*

$$\Pr\left[\begin{array}{l} T(t, L'_i) = 1 \quad \textit{for } i = 1, \ldots, p(k)+1 \\ \qquad \textit{where } (L'_1, \ldots, L'_{p(k)+1}) \leftarrow \mathit{FAKE}(L_1, \ldots, L_{p(k)}), \\ \qquad L_1 \leftarrow LG(1^k, p), \ \ldots, \ L_{p(k)} \leftarrow LG(1^k, p) \textit{ and } (p, t) \leftarrow C(1^k) \end{array}\right] < \nu(k).$$

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,*

$$\begin{aligned}
T(v, L_i) &= 1 \quad \textit{for } i = 1, \ldots, m, \quad \textit{and} \\
T(v, L'_i) &= 1 \quad \textit{for } i = 1, \ldots, n.
\end{aligned}$$

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: $\exists\, i, j \in \{1, \ldots, n\}$, $k \in \{1, \ldots, m\}$:
$M\left(\mathsf{Par}(L_k), \mathsf{Obj}(L'_i)\right) \;=\; M\left(\mathsf{Par}(L_k), \mathsf{Obj}(L'_j)\right) \;=\; M\left(\mathsf{Par}(L_k), \mathsf{Obj}(L_k)\right)$*.*

(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: $\exists\, i \in \{1, \ldots, n\} : Ver\left(v, \mathsf{Pro}(L'_i), \mathsf{Sig}(L'_i)\right) = 1$
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 $j = 1, \dots, m$. 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 $\texttt{p-}CLONE$ in $\mathcal{T}$, for all polynomials q and for all sufficiently large k,*

$$\Pr\left[\begin{array}{l} \exists\, i \neq j,\, l: \; \overline{P}_l(O'_i) = \overline{P}_l(O'_j) = \overline{P}_l(O_l) \\ \textit{and } \forall\, l\; \exists\, i: \; \overline{P}_l(O_l) = \overline{P}_l(O'_i) \\ \qquad \textit{where } (O'_1, \dots, O'_{p(k)+1}) \\ \qquad \leftarrow \texttt{p-}CLONE\left(1^k, \overline{P}_1(O_1), O_1, \dots, \overline{P}_{p(k)}(O_{p(k)}), O_{p(k)}\right) \\ \qquad \textit{and } (\overline{P}_i(O_i), O_i) \leftarrow OG(1^k) \textit{ for } i = 1, \dots, p(k) \end{array}\right] \leq 1/q(k)$$

The probability is taken over the random outputs of $\texttt{p-}CLONE$ and OG.

It can be seen easily that the notion of a p-UOS includes the notion of a UOS; indeed, a UOS is nothing more than a 2-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 "⇐" 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 "⇒" 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 φ-TM ***p-CLONE*** such that for the polynomial p from the contradiction assumption, some polynomial q and infinitely many k,

$$\Pr\left[\begin{array}{l} \exists\, i \neq j,\, l:\ \overline{P}_l(O'_i) = \overline{P}_l(O'_j) = \overline{P}_l(O_l) \\ \text{and } \forall\, l\ \exists\, i:\ \overline{P}_l(O_l) = \overline{P}_l(O'_i) \\ \qquad \text{where } (O'_1, \ldots, O'_{p(k)+1}) \\ \qquad \leftarrow \textbf{\textit{p-CLONE}}\big(1^k, \overline{P}_1(O_1), O_1, \ldots, \overline{P}_{p(k)}(O_{p(k)}), O_{p(k)}\big) \\ \qquad \text{and } \big(\overline{P}_i(O_i), O_i\big) \leftarrow OG(1^k) \ \text{ for } i = 1, \ldots, p(k) \end{array}\right] > 1/q(k),$$

where the probability is as taken over the random outputs of ***p-CLONE*** and *OG*.

The physical Turing machine ***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 ***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 φ-TM ***p-CLONE***.

```
Machine CLONE:

  Input $\big(1^k, \overline{P}(O), O\big)$

      Choose $i_0$ uniformly at random from $\{1, \ldots, p(k)\}$

      Set $\big(\overline{P}(O_{i_0}), O_{i_0})\big) \leftarrow \big(\overline{P}(O), O\big)$

      Set $\big(\overline{P}(O_i), O_i\big) \leftarrow OG(1^k)$   for $i = 1, \ldots, i_0 - 1, i_0 + 1, \ldots, p(k)$

      Set $\big(\overline{P}(O'_1), O'_1, \ldots, \overline{P}(O'_{p(n)+1}), O'_{p(n)+1}\big)$
          $\leftarrow$ p-CLONE $\big(\overline{P}(O_1), O_1, \ldots, \overline{P}(O_{p(n)}), O_{p(n)}\big)$

      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-CLONE*. To that aim, we introduce a random variable CopInd as follows:

$$\begin{aligned}
&\mathsf{CopInd}:\ D_{\mathsf{CopInd}} \longrightarrow \mathbb{N}_0\\
&(\overline{P}(O_1), O_1, \ldots, \overline{P}(O_t), O_t)\\
&\qquad\longmapsto \min \left\{ l \;\middle|\; \begin{array}{l} \exists i \neq j : \overline{P}_l(O'_i) = \overline{P}_l(O'_j) = \overline{P}_l(O_l),\\ \qquad \text{where } (\overline{P}_1(O'_1), O'_1, \ldots, \overline{P}(O'_{t+1}), O_{t+1})\\ \qquad \leftarrow \textit{p-CLONE}(\overline{P}_1(O_1), O_1, \ldots, \overline{P}(O_{t+1}), O_{t+1}) \end{array} \right\}
\end{aligned}$$

Note that CopInd is defined in terms of the random variable *p-CLONE*$(\cdot)$. As $\min(\emptyset) = 0$, it can be seen rather easily that for the polynomial p and any objects $O_1, \ldots, O_{p(k)}$,

$$\begin{aligned}
&\Pr\left[\ \mathsf{CopInd}(\overline{P}_1(O_1), O_1, \ldots, \overline{P}_{p(k)}(O_{p(k)}, O_{p(k)}) \neq 0\ \right] = \qquad (1)\\
&\quad = \Pr\left[\begin{array}{l} \exists\, i \neq j,\, l : \mathsf{Pro}(O'_i) = \mathsf{Pro}(O'_j) = \mathsf{Pro}(O_l),\\ \qquad \text{where } (\overline{P}'_1(O'_1), O'_1, \ldots, \overline{P}'_{p(k)+1}(O'_{p(k)+1}) \leftarrow \textit{p-CLONE}(O_1, \ldots, O_{p(k)}) \end{array} \right]
\end{aligned}$$

We can now calculate the success probability of *CLONE* in the following way.

$$\Pr\left[\begin{array}{l} \overline{P}(O) = \overline{P}(O_1) = \overline{P}(O_2),\\ \qquad \text{where } (\overline{P}(O), O_1, O_2) \leftarrow \textit{CLONE}(1^k, \overline{P}(O), O)\\ \qquad \text{and } O \leftarrow OG(1^k) \end{array} \right] =$$

$$= \Pr\left[\begin{array}{l} \mathsf{CopInd}(O_1, \ldots, O_{p(k)}) = i_0\\ \qquad \text{where } i_0 \leftarrow_{u.a.r.} \{1, \ldots, p(k)\},\ O_{i_0} \leftarrow OG(1^k),\\ \qquad \text{and } O_i \leftarrow OG(1^k) \text{ for } i = 1, \ldots, i_0 - 1, i_0 + 1, \ldots, p(k) \end{array} \right]$$

$$= \Pr\left[\begin{array}{l} \mathsf{CopInd}(O_1, \ldots, O_{p(k)}) = i_0\\ \qquad \text{where } i_0 \leftarrow_{u.a.r.} \{1, \ldots, p(k)\}\\ \qquad \text{and } O_i \leftarrow OG(1^k) \text{ for } i = 1, \ldots, p(k) \end{array} \right]$$

$$= \sum_{j=1}^{p(k)} \Pr\left[\begin{array}{l} \mathsf{CopInd}(O_1, \ldots, O_{p(k)}) = j \text{ and } i_0 = j\\ \qquad \text{where } i_0 \leftarrow_{u.a.r.} \{1, \ldots, p(k)\}\\ \qquad \text{and } O_i \leftarrow OG(1^k) \text{ for } i = 1, \ldots, p(k) \end{array} \right]$$

$$\begin{aligned}
= \sum_{j=1}^{p(k)} &\Pr\left[\begin{array}{l} i_0 = j\\ \qquad \text{where } i_0 \leftarrow_{u.a.r.} \{1, \ldots, p(k)\} \end{array} \right] \cdot\\
&\cdot \Pr\left[\begin{array}{l} \mathsf{CopInd}(O_1, \ldots, O_{p(k)}) = j\\ \qquad \text{where } O_i \leftarrow OG(1^k) \text{ for } i = 1, \ldots, p(k) \end{array} \right]
\end{aligned}$$

$$= \frac{1}{p(k)} \cdot \sum_{j=1}^{p(k)} \Pr\left[\begin{array}{l} \mathsf{CopInd}(O_1, \ldots, O_{p(k)}) = j\\ \qquad \text{where } O_i \leftarrow OG(1^k) \text{ for } i = 1, \ldots, p(k) \end{array} \right]$$

$$= \frac{1}{p(k)} \cdot \Pr\left[\begin{array}{l} \mathsf{CopInd}(O_1,\dots,O_{p(k)}) \neq 0 \\ \qquad \text{where } O_i \leftarrow OG(1^k) \text{ for } i = 1,\dots,p(k) \end{array}\right]$$

$$= \frac{1}{p(k)} \cdot \Pr\left[\begin{array}{l} \exists\, i \neq j,\, l : \mathsf{Pro}(O'_i) = \mathsf{Pro}(O'_j) = \mathsf{Pro}(O_l), \\ \qquad \text{where } (O'_1,\dots,O'_{p(k)+1}) \leftarrow p\text{-}CLONE(O_1,\dots,O_{p(k)+1}) \\ \qquad \text{and } O_i \leftarrow OG(1^k) \text{ for } i = 1,\dots,p(k) \end{array}\right]$$

Or, to summarize our calculation,

$$\Pr\left[\begin{array}{l} \overline{P}(O) = \overline{P}(O_1) = \overline{P}(O_2), \\ \qquad \text{where } (\overline{P}(O), O_1, O_2) \leftarrow CLONE(1^k, \overline{P}(O), O) \\ \qquad \text{and } O \leftarrow OG(1^k) \end{array}\right] = \tag{2}$$

$$= \frac{1}{p(k)} \cdot \Pr\left[\begin{array}{l} \exists\, i \neq j,\, l : \mathsf{Pro}(O'_i) = \mathsf{Pro}(O'_j) = \mathsf{Pro}(O_l), \\ \qquad \text{where } (O'_1,\dots,O'_{p(k)+1}) \leftarrow p\text{-}CLONE(O_1,\dots,O_{p(k)+1}) \\ \qquad \text{and } O_i \leftarrow OG(1^k) \text{ for } i = 1,\dots,p(k) \end{array}\right]$$

We further know from statement (1) that for infinitely many k and a polynomial q,

$$\Pr\left[\begin{array}{l} \exists\, i \neq j,\, l : \mathsf{Pro}(O'_i) = \mathsf{Pro}(O'_j) = \mathsf{Pro}(O_l), \\ \qquad \text{where } (O'_1,\dots,O'_{p(k)+1}) \leftarrow p\text{-}CLONE(O_1,\dots,O_{p(k)+1}) \\ \qquad \text{and } O_i \leftarrow OG(1^k) \text{ for } i = 1,\dots,p(k) \end{array}\right] > 1/q(k).$$

Inserting this into equation (2) we obtain that for infinitely many k a polynomial q and a polynomial q,

$$\Pr\left[\begin{array}{l} \overline{P}(O) = \overline{P}(O_1) = \overline{P}(O_2), \\ \qquad \text{where } (\overline{P}(O), O_1, O_2) \leftarrow CLONE(1^k, \overline{P}(O), O) \\ \qquad \text{and } O \leftarrow OG(1^k) \end{array}\right] > 1/p(k) \cdot 1/q(k).$$

Hence it holds that for infinitely many k and a polynomial $r \overset{\text{def}}{=} p \cdot q$,

$$\Pr\left[\begin{array}{l} \overline{P}(O) = \overline{P}(O_1) = \overline{P}(O_2), \\ \qquad \text{where } (\overline{P}(O), O_1, O_2) \leftarrow CLONE(1^k, \overline{P}(O), O) \\ \qquad \text{and } O \leftarrow OG(1^k) \end{array}\right] > 1/r(k).$$

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. □

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 $\mathsf{p\text{-}UOS}$ 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 φ-TM $FAKE$ in $\mathcal{T}$ and polynomials p, q such that for infinitely many n,

$$\Pr\left[\begin{array}{l} T(t, L'_i) = 1 \quad \text{for } i = 1, \ldots, p(n)+1 \\ \qquad \text{where } (L'_1, \ldots, L'_{p(n)+1}) \leftarrow FAKE(L_1, \ldots, L_{p(n)}), \\ \qquad L_1 \leftarrow LG(s),\ \ldots,\ L_{p(n)} \leftarrow LG(s) \quad \text{and} \quad (s, v) \leftarrow C(1^n) \end{array}\right] \geq q(n),$$

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 φ-TM. 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* φ-TM. 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 $\big(1^k, \overline{P}(O_1), O_1, \ldots, \overline{P}(O_{p(k)}), O_{p(k)}\big)$
    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 $\big(\mathsf{Obj}(L'_1), \ldots, \mathsf{Obj}(L'_{p(k)+1})\big)$
```

p-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 φ-TM *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. *p-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: *p-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 *p-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  $(L'_1,\dots,L'_{p(n)+1}) \leftarrow \mathit{FAKE}(L_1,\dots,L_{p(n)})$
        If  $\exists\, i_0 \in \{1,\dots,p(n)+1\} : \left(\begin{array}{l} \mathit{Ver}\,(v,\mathsf{Pro}(L'_{i_0}),\mathsf{Sig}(L'_{i_0}))=1 \\ \text{and } \nexists\, j \in 1,\dots,p(n) : \mathsf{Pro}(L'_{i_0})=\mathsf{Pro}(L_j) \end{array}\right)$
            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

$$\Pr\left[\begin{array}{l} T(t,L'_i)=1 \quad \text{for } i=1,\dots,p(k)+1 \\ \qquad \text{where } (L'_1,\dots,L'_{p(k)+1}) \leftarrow \mathit{FAKE}(L_1,\dots,L_{p(k)}), \\ \qquad L_1 \leftarrow LG(s),\ \dots,\ L_{p(k)} \leftarrow LG(s) \text{ and } (s,v) \leftarrow C(1^k) \end{array}\right]$$

$$= \Pr\left[\begin{array}{l} T(t,L'_i)=1 \quad \text{for } i=1,\dots,p(k)+1 \\ \text{and } T(t,L_i)=1 \quad \text{for } i=1,\dots,p(k) \\ \qquad \text{where } (L'_1,\dots,L'_{p(n)+1}) \leftarrow \mathit{FAKE}(L_1,\dots,L_{p(k)}), \\ \qquad L_1 \leftarrow LG(s),\ \dots,\ L_{p(k)} \leftarrow LG(s) \text{ and } (s,v) \leftarrow C(1^k) \end{array}\right]$$

$$= \Pr\left[\begin{array}{l} \left(\begin{array}{l} \exists\, i \neq j,\, l:\ M\big(\mathsf{Par}(L_l),\mathsf{Obj}(L'_i)\big) = \\ \quad = M\,(\mathsf{Par}(L_l),\mathsf{Obj}(L'_j)) = M\big(\mathsf{Par}(L_l),\mathsf{Obj}(L_l)\big) \end{array}\right) \\ \text{or } \left(\begin{array}{l} \exists i:\ \mathit{Ver}\big((v,\mathsf{Pro}(L'_i),\mathsf{Sig}(L'_i)\big)=1 \\ \quad \text{and } \nexists j : \mathsf{Pro}(L'_i)=\mathsf{Pro}(L_j) \end{array}\right) \\ \text{where } (L'_1,\dots,L'_{p(k)+1}) \leftarrow \mathit{FAKE}(L_1,\dots,L_{p(k)}), \\ L_1 \leftarrow LG(s),\ \dots\ ,\ L_{p(k)} \leftarrow LG(s) \text{ and } (s,v) \leftarrow C(1^k) \end{array}\right]$$

(because of Lemma 6.7)

$$\leq \Pr\left[\begin{array}{l} \exists\, i \neq j,\, l:\ M\,(\mathsf{Par}(L_l),\mathsf{Obj}(L'_i)) = \\ \quad = M\,(\mathsf{Par}(L_l),\mathsf{Obj}(L'_j)) = M\,(\mathsf{Par}(L_l),\mathsf{Obj}(L_l)) \\ \qquad \text{where } (L'_1,\dots,L'_{p(k)+1}) \leftarrow \mathit{FAKE}(L_1,\dots,L_{p(k)}), \\ \qquad L_1 \leftarrow LG(s),\dots,\ L_{p(k)} \leftarrow LG(s) \text{ and } (s,v) \leftarrow C(1^k) \end{array}\right]$$

$$+ \Pr\left[\begin{array}{l} \exists i:\ \mathit{Ver}\big((v,\mathsf{Pro}(L'_i),\mathsf{Sig}(L'_i)\big)=1 \\ \quad \text{and } \nexists j : \mathsf{Pro}(L'_i)=\mathsf{Pro}(L_j) \\ \qquad \text{where } (L'_1,\dots,L'_{p(k)+1}) \leftarrow \mathit{FAKE}(L_1,\dots,L_{p(k)}), \\ \qquad L_1 \leftarrow LG(s),\dots,\ L_{p(k)} \leftarrow LG(s) \text{ and } (s,v) \leftarrow C(1^k) \end{array}\right]$$

(because $P(A \cup B) \leq P(A)+P(B)$)

$$= \Pr\left[\begin{array}{l} \mathit{CLONE}\big(1^k,\overline{P}(O_1),O_1,\dots,\overline{P}(O_{p(k)}),O_{p(k)}\big) \neq \text{“}failure''\\ \qquad \text{where } O_i \leftarrow OG(1^k) \quad \text{for } i=1,\dots,p(k) \end{array}\right]$$

$$+ \Pr \left[\begin{array}{c} \mathit{SIGBR}\,(1^k, v) \neq \text{“}\mathit{failure}\text{''} \\ \text{where } (s, v) \leftarrow C(1^k) \end{array} \right]$$

(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,

$$\Pr \left[\begin{array}{l} \mathit{CLONE}\,(1^k, \overline{P}(O_1), O_1, \ldots, \overline{P}(O_{p(k)}), O_{p(k)}) \neq \text{“}\mathit{failure}\text{''} \\ \quad \text{where } O_i \leftarrow OG(1^k) \quad \text{for } i = 1, \ldots, p(k) \end{array} \right]$$

$$+ \Pr \left[\begin{array}{c} \mathit{SIGBR}\,(1^k, v) \neq \text{“}\mathit{failure}\text{''} \\ \text{where } (s, v) \leftarrow C(1^k) \end{array} \right]$$

$$\geq \Pr \left[\begin{array}{l} T(t, L_i') = 1 \quad \text{for } i = 1, \ldots, p(k) + 1 \\ \quad \text{where } (L_1', \ldots, L_{p(k)+1}') \leftarrow \mathit{FAKE}\,(L_1, \ldots, L_{p(k)}), \\ \quad L_1 \leftarrow LG(s),\ \ldots,\ L_{p(k)} \leftarrow LG(s) \text{ and } (s, v) \leftarrow C(1^k) \end{array} \right]$$

(by the previous calculation)

$$> 1/q(k)$$

(by statement (1)).

This implies that for infinitely many k,

$$\Pr \left[\begin{array}{l} \mathit{CLONE}\,(1^k, \overline{P}(O_1), O_1, \ldots, \overline{P}(O_{p(k)}, O_{p(k)}) \neq \text{“}\mathit{failure}\text{''} \\ \quad \text{where } O_i \leftarrow OG(1^k) \quad \text{for } i = 1, \ldots, p(k) \end{array} \right] > 1/q(k)$$

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 or φ-TMs, 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 [25, 22].

One quite obvious problem in this context is the consistent definition of a "physical generator", which could serve as an analog to the generators in classic cryptography, for example in the formalization of digital signature schemes. Such a physical generator should take as input a number of the form 1^k, and output those physical objects that are used in physical cryptography (Unique Objects, PUFs, etc.) of ever growing size depending on k. If such a generator is just said to be a finite physical *"device"* or *"machine"* — how can a finite device of finite mass produce objects of growing size, whose mass will eventually exceed the mass of the device itself? This observation poses a problem for any formalization attempts on physical cryptography. PhTMs and their inherently asymptotic nature help us to overcome these problems, however, as illustrated in this paper.

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 [22, 21, 5, 18]). 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] Arjen K. Lenstra, Adi Shamir: *Analysis and optimization of the TWINKLE factoring device.* EUROCRYPT 2000, pp. 35-52.

[16] Ueli M. Maurer: *Conditionally-perfect secrecy and a provably-secure randomized cipher.* Journal of Cryptology 5.1 (1992): 53-66.

[17] Ueli M. Maurer: *Secret key agreement by public discussion from common information.* Information Theory, IEEE Transactions on 39.3 (1993): 733-742.

[18] Rafail Ostrovsky, Alessandra Scafuro, Ivan Visconti, Akshay Wadia: *Universally Composable Secure Computation with (Malicious) Physically Uncloneable Functions.* EUROCRYPT 2013, pp. 702-718.

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

[20] Ulrich Rührmair: *SIMPL Systems: On a Public Key Variant of Physical Unclonable Functions.* IACR Cryptology ePrint Archive 2009:255 (2009).

[21] Ulrich Rührmair: *Oblivious Transfer Based on Physical Unclonable Functions.* TRUST 2010, pp. 430-440.

[22] Ulrich Rührmair, Heike Busch, Stefan Katzenbeisser: *Strong PUFs: Models, Constructions, and Security Proofs.* In: Towards Hardware-Intrinsic Security, Springer, 2010, pp. 79-96.

[23] 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

[24] Ulrich Rührmair, Marten van Dijk: *PUFs in Security Protocols: Attack Models and Security Evaluations.* IEEE Symposium on Security and Privacy 2013, pp. 286-300.

[25] Ulrich Rührmair, Jan Sölter, Frank Sehnke: *On the Foundations of Physical Unclonable Functions.* IACR Cryptology ePrint Archive 2009: 277 (2009)

[26] Adi Shamir: *"Factoring large numbers with the TWINKLE device."* CHES 2009, pp. 2-12.

[27] Andrew Chi-Chih Yao: *Classical physics and the Church-Turing Thesis.* Journal of the ACM 50(1), 100-105, 2003.