Unconditionally Secure and Universally Composable Commitments from Physical Assumptions

Ivan Damgård[1⋆] and Alessandra Scafuro[2⋆⋆]

[1] Dept. of Computer Science, Aarhus University, Denmark
[2] Dept. of Computer Science, UCLA, USA

Abstract. We present a constant-round unconditional black-box compiler that transforms any ideal (i.e., statistically-hiding and statistically-binding) straight-line extractable commitment scheme, into an extractable and equivocal commitment scheme, therefore yielding to UC-security [9]. We exemplify the usefulness of our compiler by providing two (constant-round) instantiations of ideal straight-line extractable commitment based on (malicious) PUFs [36] and *stateless* tamper-proof hardware tokens [26], therefore achieving the first unconditionally UC-secure commitment with malicious PUFs and stateless tokens, respectively. Our constructions are secure for adversaries creating arbitrarily malicious stateful PUFs/tokens.

Previous results with malicious PUFs used either computational assumptions to achieve UC-secure commitments or were unconditionally secure but only in the indistinguishability sense [36]. Similarly, with stateless tokens, UC-secure commitments are known only under computational assumptions [13,24,15], while the (not UC) unconditional commitment scheme of [23] is secure only in a weaker model in which the adversary is not allowed to create stateful tokens.

Besides allowing us to prove feasibility of unconditional UC-security with (malicious) PUFs and stateless tokens, our compiler can be instantiated with any ideal straight-line extractable commitment scheme, thus allowing the use of various setup assumptions which may better fit the application or the technology available.

Keywords: UC-security, hardware assumptions, unconditional security, commitment scheme.

⋆ The authors acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, within which part of this work was performed; and also from the CFEM research center (supported by the Danish Strategic Research Council) within which part of this work was performed.

⋆⋆ Work done while visiting Aarhus University. Research supported in part by NSF grants CCF-0916574; IIS-1065276; CCF-1016540; CNS-1118126; CNS- 1136174; and Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392.

1 Introduction

Unconditional security guarantees that a protocol is secure even when the adversary is unbounded. While it is known how to achieve unconditional security for multi-party functionalities in the plain model assuming honest majority [4,14], obtaining unconditionally secure *two-party* computation is impossible in the plain model. In fact, for all non-trivial two-party functionalities, achieving unconditional security requires some sort of (physical) setup assumption.

Universally composable (UC) security [9] guarantees that a protocol is secure even when executed concurrently with many other instances of any arbitrary protocol. This strong notion captures the real world scenarios, where typically many applications are run concurrently over the internet, and is therefore very desirable to achieve. Unfortunately, achieving UC-security in the plain model is impossible [11].

Hence, constructing 2-party protocols which are unconditionally secure or universally composable requires the employment of some setup. One natural research direction is to explore which setup assumptions suffice to achieve (unconditional) UC-security, as well as to determine whether (or to what extent) we can *reduce* the amount of trust in a third party. Towards this goal, several setup assumptions have been explored by the community.

In [12] Canetti et. al show that, under computational assumptions, any functionality can be UC-realized assuming the existence of a trusted Common Reference String (CRS). Here, the security crucially relies on the CRS being honestly sampled. Hence, security in practice would typically rely on a third party sampling the CRS honestly and security breaks down if the third party is not honest. Similar arguments apply to various assumptions like "public-key registration" services [3,10].

Another line of research explores "physical" setup assumptions. Based on various types of noisy channels, unconditionally secure Bit Commitment (BC) and Oblivious Transfer (OT) can be achieved [16,17] for two parties, but UC security has not been shown for these protocols and in fact seems non-trivial to get for the case of [17].

In [26] Katz introduces the assumption of the existence of tamper-proof hardware tokens. The assumption is supported by the possibility of implementing tamper-proof hardware using current available technology (e.g., smart cards). A token is defined as a physical device (a wrapper), on which a player can upload the code of any functionality, and the assumption is that any adversary cannot tamper with the token. Namely, the adversary has only black-box access to the token, i.e., it cannot do more then observing the input/output characteristic of the token. The main motivation behind this new setup assumption is that it allows for a *reduction of trust*. Indeed in Katz's model tokens are not assumed to be trusted (i.e., produced by a trusted party) and the adversary is allowed to create a token that implements an arbitrary malicious function instead of the function dictated by the protocol. (However, it is assumed that once the token is sent away to the honest party, it cannot communicate with its creator. This assumption is necessary, as otherwise we are back to the plain model). A con-

sequence of this model is that the security of a player now depends only on its *own token* being good and holds even if tokens used by other players are not genuine! This new setup assumptions has gained a lot of interest and several works after [26] have shown that unconditional UC-security is possible [31,24], even using a single *stateful* token [21,22]. Note that a stateful token, in contrast with a *stateless* token, requires an updatable memory that can be subject to reset attacks. Thus, ensuring tamper-proofness for a stateful token seems to be more demanding than for a stateless token, and hence having protocols working with stateless tokens is preferable.

However, the only constructions known for stateless tokens require computational assumptions [13,29,24,15] and a non-constant number of rounds (if based on one-way functions only). In fact, intuitively it seems challenging to achieve unconditional security with stateless tokens: A stateless token runs always on the same state, thus an unbounded adversary might be able to extract the secret state after having observed only a polynomial number of the token's outputs. This intuition is confirmed by [23] where it is proved that unconditional OT is impossible using stateless tokens. On the positive side, [23] shows an unconditional commitment scheme (not UC) based on stateless tokens, but the security of the scheme holds only if the adversary is not allowed to create malicious stateful tokens. This is in contrast with the standard tamper-proof hardware model, where the adversary is allowed to construct any arbitrary malicious (hence possibly stateful) token. Indeed, it seems difficult in practice to detect whether an adversary sends a stateless or a stateful token. Therefore, the question of achieving unconditional commitments (UC-secure or not) in the standard stateless token model (where an adversary possibly plays with stateful tokens) is still open.

In this work we provide a positive answer showing the first UC-secure unconditional commitment scheme with stateless tokens.

Following the approach of [26], Brzuska et al. in [7] propose a new setup assumption for achieving UC security, which is the existence of Physically Uncloneable Functions (PUFs). PUFs have been introduced by Pappu in [38,37], and since then have gained a lot of interest for cryptographic applications [2,42,1,40]. A PUF is a physical noisy source of randomness. In other words a PUF is a device implementing a function whose behavior is unpredictable even to the manufacturer. The reason is that even knowing the exact manufacturing process there are parameters that cannot be controlled, therefore it is assumed infeasible to construct two PUFs with the same challenge-response behavior. A PUF is noisy in the sense that, when queried twice with the same challenge, it can output two different, although close, responses. Fuzzy extractors are applied to PUF's responses in order to reproduce a unique response for the same challenge. The "PUF assumption" consists in assuming that PUFs satisfy two properties: 1) unpredictability: the distribution implemented by a PUF is unpredictable. That is, even after a polynomial number of challenge/response pairs have been observed, the response on any new challenge (sufficiently far from the ones observed so far) is unpredictable; this property is *unconditional*; 2) uncloneability: as a PUF is the output of a physical uncontrollable manufacturing process, it is assumed

that creating two identical PUFs is hard even for the manufacturer. This property is called hardware uncloneability. Software uncloneability corresponds to the hardness of modeling the function implemented by the PUF and is enforced by unpredictability (given that the challenge/response space of the PUF is adequately large). Determining whether (or to what extent) current PUF candidates actually satisfy the PUF assumption is an active area of research (e.g., [27,5]) but is out of the scope of this work. For a survey on PUF's candidates the reader can refer to [30], while a security analysis of silicon PUFs is provided in [27].

Designing PUF-based protocols is fundamentally different than for other hardware tokens. This is due to the fact that the functional behavior of a PUF is unpredictable even for its creator. Brzuska et al. modeled PUFs in the UC-setting by formalizing the ideal PUF functionality. They then provided constructions for Unconditional UC Oblivious Transfer and Bit Commitment. However, their UC-definition of PUFs assumes that all PUFs are *trusted*. Namely, they assume that even a malicious player creates PUFs honestly, following the prescribed generation procedure. This assumption seems too optimistic as it implies that an adversary must not be capable of constructing hardware that "looks like" a PUF but that instead computes some arbitrary function. The consequence of assuming that all PUFs are trusted is that the security of a player depends on the PUFs created by other players. (Indeed, in the OT protocol of [7], if the receiver replaces the PUF with hardware implementing some predictable function, the security of the sender is violated).

In [36] Ostrovsky et al. extend the ideal PUF functionality of [7] in order to model the adversarial behavior of creating and using "malicious PUFs". A malicious PUF is a physical device for which the security properties of a PUF are not guaranteed. As such, it can be a device implementing *any* function chosen by the adversary, so that the adversary might have full control on the answers computed by its own "PUF". Similarly to the hardware-token model, a malicious PUF cannot communicate with the creator once is sent away. A malicious PUF can, of course, be stateful. The major advantage of the malicious PUF model is that the security of a player depends only on the goodness of its own PUFs. Obviously, the price to pay is that protocols secure in presence of malicious PUFs are more complex than protocols designed to deal only with honest PUFs. Nevertheless, [36] shows that even with malicious PUFs it is possible to achieve UC-secure computations relying on computational assumptions. They also show an unconditional commitment scheme which is secure only in the indistinguishability sense. Achieving unconditional UC-secure commitments (and general secure computations) is left as an open problem in [36].

In this paper, we give a (partial) positive answer to this open problem by providing the first construction of unconditional UC-secure Bit Commitment in the malicious PUFs model. Whether unconditional OT (and thus general secure computation) is possible with malicious PUFs is still an interesting open question. Intuitively, since PUFs are stateless devices, one would think to apply the arguments used for the impossibility of unconditional OT with stateless tokens [23]. However, due to the unpredictability property of PUFs which holds

unconditionally, such arguments do not carry through. Indeed, as long as there is at least one honest PUF in the system, there is enough entropy, and this seems to defeat the arguments used in [23]. On the other hand, since a PUF is in spirit just a "random function", it is not clear how to implement the OT functionality when only one of the players uses honest PUFs.

Van Dijk and Rührmair in [19] also consider adversaries who create malicious PUFs, that they call "bad PUFs" and they consider only the stand-alone setting. They show that unconditional OT is impossible in the bad PUF model but this impossibility proof works assuming that also honest parties play with bad PUFs. Thus, such impossibility proof has no connection to the question of achieving unconditional OT in the malicious PUF model (where honest parties play with honest PUFs).

Our Contribution. In this work we provide a tool for constructing UC-secure commitments given any straight-line extractable commitment. This tool allows us to show feasibility results for unconditional UC-secure protocols in the stateless token model and in the malicious PUF model. More precisely, we provide an unconditional black-box compiler that transforms any ideal (i.e., statistically hiding and binding) straight-line extractable commitment into a UC-secure commitment. The key advantage of such compiler is that one can implement the ideal extractable commitment with the setup assumption that is more suitable to the application and the technology available.

We then provide an implementation of the ideal extractable commitment scheme in the malicious PUFs model of [36]. Our construction builds upon the (stand-alone) unconditional BC scheme shown in [36] [3] which is not extractable. By plugging our extractable commitment scheme in our compiler we obtain the first unconditional UC-secure commitment with malicious PUFs.

We then construct ideal extractable commitments using stateless tokens. We use some of the ideas employed for the PUF construction, but implement them with different techniques. Indeed, while PUFs are intrinsically unpredictable and even having oracle access to a PUF an unbounded adversary cannot predict the output on a new query, with stateless tokens we do not have such guarantee. Our protocol is secure in the standard token model, where the adversary has no restriction and can send malicious stateful tokens. By plugging such protocol in our compiler, we achieve the first unconditional UC-secure commitment scheme with stateless tokens. Given that unconditional OT is impossible with stateless tokens, this result completes the picture concerning feasibility of unconditional UC-security in this model.

Related Work. Our compiler can be seen as a generalization of the black-box trapdoor commitment given by Pass and Wee [39] which is secure only in the

[3] For completeness, we would like to mention that [41] claims an "attack" on such construction. Such "attack" however arises only due to misunderstanding of conventions used to write protocol specifications and does not bear any security threat. The reader can refer to the discussion of [35] (full version of [36]) at Pag. 7, paragraph "On [RvD13]", line 20–40 for more details.

computational stand-alone setting. Looking ahead to our constructions of extractable commitment, the idea of querying the hardware token with the opening of the commitment was first used by Müller-Quade and Unruh in [32,33], and later by Chandran et al. in [13]. The construction of [13] builds UC-secure multiple commitments on top of extractable commitments. Their compiler requires computational assumptions, logarithmic number of rounds and crucially uses cryptographic primitives in a non-black box manner.

Remark 1. In the rest of the paper it is assumed that even an unbounded adversary can query the PUF/token only a polynomial number of times. We stress that this is not a restriction of our work but it is a necessary assumption in all previous works achieving unconditional security with PUFs and stateless tokens (see pag.15 of [8] for PUFs, and pag. 5 of [23] for stateless tokens). Indeed, if we allowed the adversary to query the PUF/token on all possible challenges, then she can derive the truth table implemented by the physical device.

2 Definitions

Notation. We denote the security parameter by n, and the property of a probabilistic algorithm whose number of steps is polynomial in its security parameter, by PPT. We denote by $(v_A, v_B) \leftarrow \langle A(a), B(b)\rangle(x)$ the local outputs of A and B of the random process obtained by having A and B activated with independent random tapes, interacting on common input x and on (private) auxiliary inputs a to A and b to B. When the common input x is the security parameter, we omit it. If A is a probabilistic algorithm we use $v \xleftarrow{\$} A(x)$ to denote the output of A on input x assigned to v. We denote by $\mathsf{view}_A(A(a), B(b))(x)$ the view of A of the interaction with player B, i.e., its values is the transcript $(\gamma_1, \gamma_2, ..., \gamma_t; r)$, where the γ_i's are all the messages exchanged and r is A's coin tosses. We use notation $[n]$ to denote the set $\{1, \ldots, n\}$. Let P_1 and P_2 be two parties running protocol (A,B) as sub-routine. When we say that party "P_1 runs $\langle A(\cdot), B(\cdot)\rangle(\cdot)$ with P_2" we always mean that P_1 executes the procedure of party A and P_2 executes the procedure of party B. In the following definitions we assume that the adversary has auxiliary information, and assume that parties are stateful.

2.1 Ideal Extractable Commitment Scheme

We denote by $\mathcal{F}_{\mathsf{aux}}$ an auxiliary set-up functionality accessed by the real world parties (and by the extractor).

Definition 1 (Ideal Commitment Scheme in the $\mathcal{F}_{\mathsf{aux}}$-hybrid model). *A commitment scheme is a tuple of PPT algorithms* $\mathsf{Com} = (\mathsf{C}, \mathsf{R})$ *having access to an ideal set-up functionality* $\mathcal{F}_{\mathsf{aux}}$*, implementing the following two-phase functionality. Given to* C *an input* $b \in \{0,1\}$*, in the first phase called* commitment phase*,* C *interacts with* R *to commit to the bit* b*. We denote this interaction by* $((\mathbf{c}, d), \mathbf{c}) \leftarrow \langle \mathsf{C}(\mathtt{com}, b), \mathsf{R}(\mathtt{recv})\rangle$ *where* $\mathbf{c}$ *is the transcript of the (possibly interactive) commitment phase and* d *is the decommitment data. In the second*

phase, called decommitment phase, C *sends* (b, d) *and* R *finally outputs "*accept*" or "*reject*" according to* $(\mathbf{c}, d, b)$*. In both phases parties could access to* $\mathcal{F}_{\mathsf{aux}}$*.* $\mathsf{Com} = (\mathsf{C}, \mathsf{R})$ *is an* ideal *commitment scheme if it satisfies the following properties.*

- **Completeness.** *For any* $b \in \{0,1\}$*, if* C *and* R *follow their prescribed strategy then* R *accepts the commitment* $\mathbf{c}$ *and the decommitment* (b, d) *with probability 1.*
- **Statistically Hiding.** *For any malicious receiver* R^* *the ensembles* $\{\mathsf{view}_{\mathsf{R}^*}(\mathsf{C}(\mathtt{com}, 0), \mathsf{R}^*)\ (1^n)\}_{n\in\mathbb{N}}$ *and* $\{\mathsf{view}_{\mathsf{R}^*}(\mathsf{C}(\mathtt{com}, 1), \mathsf{R}^*)\ (1^n)\}_{n\in\mathbb{N}}$ *are statistically indistinguishable, where* $\mathsf{view}_{\mathsf{R}^*}(\mathsf{C}(\mathtt{com}, b), \mathsf{R}^*)$ *denotes the view of* R^* *restricted to the commitment phase.*
- **Statistically Binding.** *For any malicious committer* C^**, there exists a negligible function* ϵ*, such that* C^* *succeeds in the following game with probability at most* $\epsilon(n)$*: On security parameter* 1^n*,* C^* *interacts with* R *in the commitment phase obtaining the transcript* $\mathbf{c}$ *. Then* C^* *outputs pairs* $(0, d_0)$ *and* $(1, d_1)$*, and succeeds if in the decommitment phase,* $\mathsf{R}(\mathbf{c}, d_0, 0) = \mathsf{R}(\mathbf{c}, d_1, 1) = \mathsf{accept}$.

In this paper the term *ideal* is used to refer to a commitment which is statistically-hiding and statistically-binding.

Definition 2 (Interface Access to an Ideal Functionality $\mathcal{F}_{\mathsf{aux}}$). *Let* $\Pi = (P_1, P_2)$ *be a two-party protocol in the* $\mathcal{F}_{\mathsf{aux}}$*-hybrid model. That is, parties* P_1 *and* P_2 *need to query the ideal functionality* $\mathcal{F}_{\mathsf{aux}}$ *in order to carry out the protocol. An algorithm* M *has* interface access *to the ideal functionality* $\mathcal{F}_{\mathsf{aux}}$ *w.r.t. protocol* Π *if all queries made by either party* P_1 *or* P_2 *to* $\mathcal{F}_{\mathsf{aux}}$ *during the protocol execution are observed (but not answered) by* M*, and* M *has oracle access to* $\mathcal{F}_{\mathsf{aux}}$*. Consequently,* $\mathcal{F}_{\mathsf{aux}}$ *can be a non programmable and non PPT functionality.*

Definition 3 (Ideal Extractable Commitment Scheme in the $\mathcal{F}_{\mathsf{aux}}$ model). $\mathsf{IdealExtCom} = (\mathsf{C}_{\mathsf{ext}}, \mathsf{R}_{\mathsf{ext}})$ *is an* ideal extractable commitment scheme in the $\mathcal{F}_{\mathsf{aux}}$ model *if* $(\mathsf{C}_{\mathsf{ext}}, \mathsf{R}_{\mathsf{ext}})$ *is an ideal commitment and there exists a straight-line strict polynomial-time extractor* E *having* interface access *to* $\mathcal{F}_{\mathsf{aux}}$*, that runs the commitment phase only and outputs a value* $b^\star \in \{0, 1, \perp\}$ *such that, for all malicious committers* C^**, the following properties are satisfied.*

- **Simulation:** *the view generated by the interaction between* E *and* C^* *is identical to the view generated when* C^* *interacts with the honest receiver* $\mathsf{R}_{\mathsf{ext}}$*:* $\mathsf{view}^{\mathcal{F}_{\mathsf{aux}}}_{\mathsf{C}^*}(\mathsf{C}^*(\mathtt{com}, \cdot), \mathsf{R}_{\mathsf{ext}}(\mathtt{recv})) \equiv \mathsf{view}^{\mathcal{F}_{\mathsf{aux}}}_{\mathsf{C}^*}(\mathsf{C}^*(\mathtt{com}, \cdot), E)$
- **Extraction:** *let* $\mathbf{c}$ *be a valid transcript of the commitment phase run between* C^* *and* E*. If* E *outputs* $\perp$ *then probability that* C^* *will provide an accepting decommitment is negligible.*
- **Binding:** *if* $b^\star \neq \perp$*, then probability that* C^* *decommits to a bit* $b \neq b^\star$ *is negligible.*

2.2 Physically Uncloneable Functions

Here we recall the definition of PUFs taken from [7]. A Physically Uncloneable Function (PUF) is a noisy physical source of randomness. A PUF is evaluated with a physical stimulus, called the *challenge*, and its physical output, called the *response*, is measured. Because the processes involved are physical, the function implemented by a PUF can not necessarily be modeled as a mathematical function, neither can be considered computable in PPT. Moreover, the output of a PUF is noisy, namely, querying a PUF twice with the same challenge, could yield to different outputs.

A PUF-family $\mathcal{P}$ is a pair of (not necessarily efficient) algorithms Sample and Eval. Algorithm Sample abstracts the PUF fabrication process and works as follows. Given the security parameter in input, it outputs a PUF-index id from the PUF-family satisfying the security property (that we define soon) according to the security parameter. Algorithm Eval abstracts the PUF-evaluation process. On input a challenge s, it evaluates the PUF on s and outputs the response σ. A PUF-family is parametrized by two parameters: the bound on the noisy of the answers d_{noise}, and the size of the range rg. A PUF is assumed to satisfy the bounded noise condition, that is, when running $\mathsf{Eval}(1^n, \mathsf{id}, s)$ twice, the Hamming distance of any two responses σ_1, σ_2 is smaller than $d_{\mathsf{noise}}(n)$. We assume that the challenge space of a PUF is the set of strings of a certain length.

Security Properties. We assume that PUFs enjoy the properties of *uncloneability* and *unpredictability*. Unpredictability is modeled in [7] via an entropy condition on the PUF distribution. Namely, given that a PUF has been measured on a polynomial number of challenges, the response of the PUF evaluated on a new challenge has still a significant amount of entropy. The following definition automatically implies uncloneability (see [8] pag. 39 for details).

Definition 4 (Unpredictability). *A (rg, d_{noise})-PUF family $\mathcal{P} = (\mathsf{Sample}, \mathsf{Eval})$ for security parameter n is $(d_{\mathsf{min}}(n), m(n))$-unpredictable if for any $s \in \{0,1\}^n$ and challenge list $\mathcal{Q} = (s_1, \ldots, s_{\mathtt{poly}(n)})$, one has that, if for all $1 \leq k \leq \mathtt{poly}(n)$ the Hamming distance satisfies $\mathsf{dis}_{\mathsf{ham}}(s, s_k) \geq d_{\mathsf{min}}(n)$, then the average min-entropy satisfies $\tilde{H}_\infty(\mathsf{PUF}(s)|\mathsf{PUF}(\mathcal{Q})) \geq m(n)$, where $\mathsf{PUF}(\mathcal{Q})$ denotes a sequence of random variables $\mathsf{PUF}(s_1), \ldots, \mathsf{PUF}(s_{\mathtt{poly}(n)})$ each one corresponding to an evaluation of the PUF on challenge s_k. Such a PUF-family is called a $(rg, d_{\mathsf{noise}}, d_{\mathsf{min}}, m)$- PUF family.*

Fuzzy Extractors. The output of a PUF is noisy. That is, querying the PUF twice with the same challenge, one might obtain two distinct responses σ, σ', that are at most d_{noise} apart in hamming distance. Fuzzy extractors of Dodis et al. [20] are applied to the outputs of the PUF, to convert such noisy, high-entropy measurements into *reproducible* randomness. Very roughly, a fuzzy extractor is a pair of efficient randomized algorithms $(\mathsf{FuzGen}, \mathsf{FuzRep})$, and it is applied to PUFs 'responses as follows. FuzGen takes as input an ℓ-bit string, that is the

PUF's response σ, and outputs a pair (p, st), where st is a uniformly distributed string, and p is a public helper data string. FuzRep takes as input the PUF's noisy response σ' and the helper data p and generates the very same string st obtained with the original measurement σ.

The security property of fuzzy extractors guarantees that, if the min-entropy of the PUF's responses are greater than a certain parameter m, knowledge of the public data p only, without the measurement σ, does not give any information on the secret value st. The correctness property, guarantees that all pairs of responses σ, σ' that are close enough, i.e., their hamming distance is less then a certain parameter t, will be recovered by FuzRep to the same value st generated by FuzGen. In order to apply fuzzy extractors to PUF's responses it is sufficient to pick an extractor whose parameters match with the parameter of the PUF being used. For formal definitions of fuzzy extractors and PUFs the reader is referred to the full version [18].

Ideal Functionalities for Malicious PUFs and Stateless Tokens. We follow the malicious PUF model introduced in [36]. In this model, the adversary is allowed to create arbitrarily malicious PUFs. Very informally, a malicious PUF is any physical device that "looks like" a PUF but it implements an arbitrary malicious, possibly stateful, function. Such adversarial behaviour has been modeled in [36] by extending the ideal functionality proposed in [7]. We denote by $\mathcal{F}_{\mathsf{PUF}}$ the ideal functionality for malicious PUF. A stateless token is a wrapper that can be programmed with any arbitrary stateless function. Tokens are modeled by [26,13] as the ideal functionality $\mathcal{F}_{\mathsf{wrap}}$. For lack of space, the formal definitions of the functionalities $\mathcal{F}_{\mathsf{PUF}}$ and $\mathcal{F}_{\mathsf{wrap}}$ together with the UC-definition are provided in the full version [18].

3 The Compiler

In this section we show how to transform any ideal extractable commitment scheme into a protocol that UC-realizes the $\mathcal{F}_{\mathsf{com}}$ functionality, unconditionally. Such transformation is based on the following building blocks.

Extractable Blobs. "Blob" was used in [6] to denote a commitment. In this paper a blob is a *pair* of bit commitments such that the actual bit committed in the blob is the xor of the pair. The representation of a bit as its exclusive-or allows to prove equality of the bits committed in two blobs using commitments as black boxes. Let IdealExtCom be any ideal extractable commitment scheme satisfying Def. 3. If the commitment phase of IdealExtCom is interactive then the blob is the pair of transcripts obtained from the interaction. Procedures to create a blob of a bit b, and to reveal the bit committed in the blob, are the following.

Blob(b): Committer picks bits b^0, b^1 uniformly at random such that $b = b^0 \oplus b^1$. It commits to b^0, b^1 (in parallel) running IdealExtCom as sub-routine and obtains commitment transcripts $\mathbf{c}^0, \mathbf{c}^1$, and decommitments d^0, d^1. Let $\mathbf{B} = (\mathbf{c}^0, \mathbf{c}^1)$ be the **blob** of b.

OpenBlob($\mathbf{B}$): Committer sends $(b^0, d^0), (b^1, d^1)$ to Receiver. Receiver accepts iff d^0, d^1 are valid decommitments of b^0, b^1 w.r.t. transcripts $(\mathbf{c}^0, \mathbf{c}^1)$ and computes $b = b^0 \oplus b^1$.

A blob inherits the properties of the commitment scheme used as sub-protocol. In particular, since IdealExtCom is used as sub-routine, each blob is statistically hiding/binding and straight-line extractable.

Equality of Blobs. Given the representation of a bit commitment as a blob, a protocol due to Kilian [28] allows to prove that two committed bits (two blobs) are equal, without revealing their values. We build upon this protocol to construct a "simulatable" version, meaning that (given some trapdoor) a simulator can prove equality of two blobs that are *not* equal. Let $\mathbf{B}_i, \mathbf{B}_j$ be two blobs. Let $b_i = (b_i^0 \oplus b_i^1)$ be the bit committed in $\mathbf{B}_i$, and $b_j = (b_j^0 \oplus b_j^1)$ be the bit committed in $\mathbf{B}_j$. Let P denote the prover and V the verifier. In the following protocol P proves to V that $\mathbf{B}_i$ and $\mathbf{B}_j$ are the commitment of the same bit (i.e., $b_i = b_j$).

BobEquality($\mathbf{B}_i, \mathbf{B}_j$)

1. V uniformly chooses $e \in \{0, 1\}$ and commits to e using IdealExtCom.
2. P sends $y = b_i^0 \oplus b_j^0$ to V.
3. V reveals e to P.
4. P reveals b_i^e and b_j^e (i.e., P sends decommitments d_i^e, d_j^e to V). V accepts iff $y = b_i^e \oplus b_j^e$.

Protocol BobEquality satisfies the following properties. **Soundness:** if $b_i \neq b_j$, any malicious prover P^* convinces V with probability negligibly close to $1/2$, that is the probability of guessing the challenge e. Here we are using the statistically hiding property of the ideal commitment IdealExtCom used to commit e. **Privacy:** If $b_i = b_j$ then after executing the protocol, the view of any verifier V^*, is independent of the actual value of b_i, b_j (given that $\mathbf{B}_i, \mathbf{B}_j$ were secure at the beginning of the protocol). **Simulation:** there exists a straight-line strictly PPT simulator SimFalse such that, for any $(\mathbf{B}_i, \mathbf{B}_j)$ that are not equal (i.e., $b_i \neq b_j$), for any malicious verifier V^*, produces a view that is statistically close to the case in which $(\mathbf{B}_i, \mathbf{B}_j)$ are equal (i.e., $b_i = b_j$) and V^* interacts with the honest P. The above properties are formally proved in the full version [18]. Note that the protocol uses blobs in a black-box way. Note also, that a blob can be involved in a single proof only.

3.1 Unconditional UC-secure Commitments from Ideal Extractable Commitments

We construct unconditional UC-secure commitments using *extractable* blobs and protocol BobEquality as building blocks. We want to implement the following idea. The committer sends two blobs of the same bit and proves that they are equal running protocol BobEquality. In the decommitment phase, it opens only one blob (a similar technique is used in [25], where instead the commitment

scheme is crucially used in a non black-box way). The simulator extracts the bit of the committer by exploiting the extractability property of blobs. It equivocates by committing to the pair 0, 1 and cheating in the protocol BobEquality. In the opening phase, it then opens the blob corresponding to the correct bit. Because soundness of BobEquality is only 1/2 we amplify it via parallel repetition.

Specifically, the committer will compute n pairs of (extractable) blobs. Then it proves equality of each pair of blobs by running protocol BobEquality with the receiver. The commitment phase is successful if all equality proofs are accepting. In the decommitment phase, the committer opens one blob for each pair. The receiver accepts if the committer opens one blob for each consecutive pair and all revealed blobs open to the same bit. The construction is formally described in Fig. 1.

Protocol UCComCompiler

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

Commitment Phase

1. Committer: run $\mathsf{Blob}(b)$ $2n$ times. Let $\mathbf{B}_1, \ldots, \mathbf{B}_{2n}$ be the **blobs** obtained.
2. Committer $\Leftrightarrow$ Receiver: for $i = 1$; $i = i + 2$; $i \leq 2n - 1$; run $\mathsf{BobEquality}(\mathbf{B}_i, \mathbf{B}_{i+1})$.
3. Receiver: if all equality proofs are accepting, accept the commitment phase.

Decommitment Phase

1. Committer: for $i = 1$; $i = i + 2$; $i \leq 2n - 1$; randomly choose $\ell \in \{i, i+1\}$ and run $\mathsf{OpenBlob}(\mathbf{B}_\ell)$ with the Receiver.
2. Receiver: 1) check if Committer opened one blob for each consecutive pair; 2) if all n blobs open to the same bit b, output b and accept. Else output reject.

Fig. 1. UCComCompiler: Unconditional UC Commitment from any Ideal Extractable Commitment.

Theorem 1. *If* IdealExtCom *is an ideal extractable commitment scheme in the* $\mathcal{F}_{\mathsf{aux}}$*-hybrid model, then protocol in Fig. 1 is an unconditionally UC-secure bit commitment scheme in the* $\mathcal{F}_{\mathsf{aux}}$*-hybrid model.*

Proof Sketch. To prove UC-security we have to show a straight-line simulator Sim which correctly simulates the view of the real-world adversary $\mathcal{A}$ and extracts her input. Namely, when simulating the malicious committer in the ideal world, Sim internally runs the real-world adversarial committer $\mathcal{A}$ simulating the honest receiver to her, so to extract the bit committed to by $\mathcal{A}$, and play it in the ideal world. This property is called extractability. When simulating the malicious receiver in the ideal world, Sim internally runs the real-world adversarial receiver $\mathcal{A}$ simulating the honest committer to her, without knowing the secret bit to commit to, but in such a way that it can be opened as any bit. This property is

called equivocality. In the following, we briefly explain why both properties are achieved.

Straight-line Extractability. It follows from the straight-line extractability and binding of IdealExtCom and from the soundness of protocol BobEquality. Roughly, Sim works as follows. It plays the commitment phase as an honest receiver (and running the straight-line extractor of IdealExtCom having access to $\mathcal{F}_{\mathsf{aux}}$). If all proofs of BobEquality are *successful*, Sim extracts the bits of each consecutive pair of blobs and analyses them as follows. Let $b \in \{0,1\}$. If all extracted pairs of bits are either (b,b) or $(\bar{b},b)$, (i.e. there are no pairs like $(\bar{b},\bar{b})$), it follows that, the only bit that $\mathcal{A}$ can successfully decommit to, is b. In this case, Sim plays the bit b in the ideal world. If there is at least a pair (b,b) and a pair $(\bar{b},\bar{b})$, then $\mathcal{A}$ cannot provide any accepting decommitment (indeed, due to the binding of blobs, $\mathcal{A}$ can only open the bit b from one pair, and the bit $\bar{b}$ from another pair, thus leading the receiver to reject). In this case Sim sends a random bit to the ideal functionality. If all the pairs of blobs are not equal, i.e., all pairs are either $(\bar{b},b)$ or $(b,\bar{b})$, then $\mathcal{A}$ can later decommit to any bit. In this case the simulator fails in the extraction of the bit committed, and it aborts. Note that, this case happens only when *all* the pairs are not equal. Thus $\mathcal{A}$ was able to cheat in all executions of BobEquality. Due to the soundness of BobEquality, this event happens with probability negligible close to 2^{-n}.

Straight-line Equivocality. It follows from the simulation property of BobEquality. Sim prepares n pairs of not equal blobs. Then it cheats in all executions of BobEquality, by running the straight-line simulator associated to this protocol. In the decommitment phase, after having received the bit to decommit to, for each pair, Sim reveals the blob corresponding to the correct bit.

Note that, in both cases Sim crucially uses the extractor associated to IdealExtCom, that in turn uses the access to $\mathcal{F}_{\mathsf{aux}}$. The formal proof of the above theorem can be found in the full version [18].

In Section 4 we show an implementation of IdealExtCom with malicious PUFs, while in Section 5, we show how to implement IdealExtCom using stateless token. By plugging such implementations in protocol UCComCompiler we obtain the first unconditional UC-secure commitment scheme with malicious PUFs (namely, in the $\mathcal{F}_{\mathsf{PUF}}$-hybrid model), and stateless tokens (namely, in the $\mathcal{F}_{\mathsf{wrap}}$-hybrid model).

4 Ideal Extractable Commitment from (Malicious) PUFs

In this section we show a construction of ideal extractable commitment in the $\mathcal{F}_{\mathsf{PUF}}$ model. Our construction builds upon the ideal commitment scheme presented in [36]. For simplicity, in the informal description of the protocol we omit mentioning the use of fuzzy extractors.

Ideal Commitment Scheme in the $\mathcal{F}_{\mathsf{PUF}}$ Model (from [36]). The idea behind the protocol of [36], that we denote by $\mathsf{CPuf} = (\mathsf{C}_{\mathsf{CPuf}}, \mathsf{R}_{\mathsf{CPuf}})$, is to turn Naor's commitment scheme [34] which is statistically binding but only computationally hiding, into statistically hiding and binding, by replacing the PRG with a (possibly *malicious*) PUF. Roughly, protocol CPuf goes as follows. At the beginning of the protocol, the committer creates a PUF, that we denote by $\mathcal{P}_S$. It preliminary queries $\mathcal{P}_S$ with a random string s (of n bits) to obtain the response σ_S (of $rg(3n)$ bits, where rg is the PUF's range) and finally sends the PUF $\mathcal{P}_S$ to the receiver. After receiving the PUF, the receiver sends a random string r (i.e., the first round of Naor's commitment) to the committer. To commit to a bit b, the committer sends $\mathbf{c} = \sigma_S \oplus (r \wedge b^{|r|})$ to the receiver. In the decommitment phase, the committer sends (b, s) to the receiver, who checks the commitment by querying $\mathcal{P}_S$ with s. For the formal description of CPuf the reader can refer to [36] or to the full version of this work [18].

Our Ideal Extractable *Commitment Scheme in the $\mathcal{F}_{\mathsf{PUF}}$ Model.* We transform CPuf into a *straight-line extractable* commitment using the following technique. We introduce a new PUF $\mathcal{P}_R$, sent by the receiver to the committer at the beginning of the protocol. Then we force the committer to query the PUF $\mathcal{P}_R$ with the opening of the commitment computed running CPuf. An opening of protocol CPuf is the value σ_S[4]. This allows the extractor, who has access to the interface of $\mathcal{F}_{\mathsf{PUF}}$, to extract the opening. The idea is that, from the transcript of the commitment (i.e., the value $\mathbf{c} = \sigma_S \oplus (r \wedge b)$) and the queries made to $\mathcal{P}_R$, (the value σ_S) the bit committed if fully determined.

To force the committer to query $\mathcal{P}_R$ with the correct opening, we require that it commits to the answer σ_R obtained by $\mathcal{P}_R$. Thus, in the commitment phase, the committer runs two instances of CPuf. One instance, that we call ComBit, is run to commit to the secret bit b. The other instance, that we call $\mathsf{ComResp}$, is run to commit to the response of PUF $\mathcal{P}_R$, queried with the opening of ComBit. In the decommitment phase, the receiver gets $\mathcal{P}_R$ back, along with the openings of both the bit and the PUF-response. Then it queries $\mathcal{P}_R$ with the opening of ComBit, and checks if the response is consistent with the string committed in $\mathsf{ComResp}$. Due to the unpredictability of PUFs, the committer cannot guess the output of $\mathcal{P}_R$ on the string σ_S without querying it. Due to the statistically binding of CPuf, the committer cannot postpone querying the PUF in the decommitment phase. Thus, if the committer will provide a valid decommitment, the extractor would have observed the opening already in the commitment phase with all but negligible probability.

However, there is one caveat. The unpredictability of PUFs is guaranteed only for queries that are sufficiently apart from each other. Which means that, given a challenge/response pair (c, r), the response on any strings c' that is "close" in hamming distance to c ("close" means that $\mathsf{dis}_{\mathsf{ham}}(c, c') \leq d_{\mathsf{min}}$), could be predictable. Consequently, a malicious committer could query the PUF with

[4] In the actual implementation we require the committer to query $\mathcal{P}_R$ with the output of the fuzzy extractor st_S, i.e., $(st_S, p_S) \leftarrow \mathsf{FuzGen}(\sigma_S)$.

a string that is only "close" to the opening. Then, given the answer to such a query, she could predict the answer to the actual opening, *without* querying the PUF. Hence, the extraction fails.

We overcome this problem by using Error Correcting Codes, in short ECC. The property of an ECC with distance parameter dis, is that any pair of strings having hamming distance dis, decodes to a unique string. Therefore, we modify the previous approach asking the committer to query PUF $\mathcal{P}_R$ with the *encoding* of the opening, i.e., $\mathsf{Encode}(\sigma_S)$. In this way, all queries that are "too close" in hamming distance decode to the same opening, and the previous attack is defeated. Informally, hiding and biding follow from hiding and binding of CPuf. Extractability follows from the statistically biding of CPuf, the unpredictability of $\mathcal{P}_R$ and the properties of ECC. The protocol is formally described in Fig. 2. In the full version [18] we discuss the parameters of the PUF and how to prevent the replacement of a honest PUF by the adversary, and we provide the proof of the following theorem.

Theorem 2. *If* CPuf *is an Ideal Commitment in the* $\mathcal{F}_{\mathsf{PUF}}$*-model, then* ExtPuf *is an Ideal* Extractable *Commitment in the* $\mathcal{F}_{\mathsf{PUF}}$ *model.*

5 Ideal Extractable Commitments from Stateless Tokens

In this section we show how to construct ideal extractable commitments from stateless tokens. We first construct an ideal commitment scheme. Then, we use it as building block for constructing an ideal *extractable* commitment.

Ideal Commitment Scheme in the $\mathcal{F}_{\mathsf{wrap}}$ *Model.* Similarly to the construction with PUFs, we implement Naor's commitment scheme by replacing the PRG with a stateless token.

In the construction with PUFs, the PRG was replaced with a PUF that is inherently unpredictable. Now, we want to achieve statistically hiding using *stateless* token. The problem here is that we do not have unpredictability for free (as it happens with PUFs). Thus, we have to program the stateless token with a function that is, somehow, unconditionally unpredictable. Clearly, we cannot construct a token that implements a PRG. Indeed, after observing a few pairs of input/output, an unbounded receiver can extract the seed, compute all possible outputs, and break hiding. We use a point function following [23] . A point function f is a function that outputs always zero, except in a particular point x, in which it outputs a value y. Formally, $f : \{0,1\}^n \to \{0,1\}^m$ such that $f(x) = y$ and $f(x') = 0$, for all $x' \neq x$.

Thus, we adapt Naor's commitment scheme as follows. The committer picks a n-bit string x and a $3n$-bit string y and creates a stateless token that on input x outputs y, while it outputs 0 on any other input. The stateless token is sent to the receiver at the beginning of the protocol. After obtaining the token, receiver sends the Naor's first message, i.e., the random value r, to the committer. The committer commits to the bit b by sending $\mathbf{c} = y \oplus (r \wedge b^{|r|})$. In the decommitment

Protocol ExtPuf

$\mathsf{ECC} = (\mathsf{Encode}, \mathsf{Decode})$ is a $(N, L, d^1_{\mathsf{min}})$ error correcting code, where $L = \ell = 3n$. Parties use PUF family: $\mathcal{P}^1 = (rg^1, d^1_{\mathsf{noise}}, d^1_{\mathsf{min}}, m^1)$, with challenge size L. $(\mathsf{FuzGen}^1, \mathsf{FuzRep}^1)$ is a $(m^1, \ell^1, t^1, \epsilon^1)$-fuzzy extractor of appropriate matching parameters. Protocol $\mathsf{CPuf} = (\mathsf{C}_{\mathsf{CPuf}}, \mathsf{R}_{\mathsf{CPuf}})$ is run as sub-routine. Committer's Input: $b \in \{0,1\}$.

Commitment Phase

1. Receiver $\mathsf{R}_{\mathsf{ExtPuf}}$: create PUF $\mathcal{P}_R$ and send it to $\mathsf{C}_{\mathsf{ExtPuf}}$.
2. **Commitment of the Secret Bit:** ComBit.
   $\mathsf{C}_{\mathsf{ExtPuf}} \Leftrightarrow \mathsf{R}_{\mathsf{ExtPuf}}$: run $\langle \mathsf{C}_{\mathsf{CPuf}}(\mathtt{com}, b), \mathsf{R}_{\mathsf{CPuf}}(\mathtt{com}) \rangle$ so that $\mathsf{C}_{\mathsf{ExtPuf}}$ commits to bit b. Let $(st_S, p_S) \leftarrow \mathsf{FuzGen}(\sigma_S)$ be the value obtained by $\mathsf{C}_{\mathsf{ExtPuf}}$, after applying the fuzzy extractor to the answer obtained from its own PUF $\mathcal{P}_S$ when running protocol ComBit.
3. Committer $\mathsf{C}_{\mathsf{ExtPuf}}$: Query $\mathcal{P}_R$ with $\mathsf{Encode}(st_S)$ and obtain response σ_R. If $\sigma_R = \perp$ (i.e., PUF $\mathcal{P}_R$ aborts), set $\sigma_R \leftarrow 0$. Compute $(st_R, p_R) \leftarrow \mathsf{FuzGen}^1(\sigma_R)$.
4. **Commitment of $\mathcal{P}_R$'s Response:** ComResp.
   $\mathsf{C}_{\mathsf{ExtPuf}} \Leftrightarrow \mathsf{R}_{\mathsf{ExtPuf}}$: run $\langle \mathsf{C}_{\mathsf{CPuf}}(\mathtt{com}, st_R || p_R), \mathsf{R}_{\mathsf{CPuf}}(\mathtt{com}) \rangle$ so that $\mathsf{C}_{\mathsf{ExtPuf}}$ commits to the string $st_R || p_R$.

Decommitment Phase

1. $\mathsf{C}_{\mathsf{ExtPuf}} \Leftrightarrow \mathsf{R}_{\mathsf{ExtPuf}}$: run $\langle \mathsf{C}_{\mathsf{CPuf}}(\mathtt{open}, b), \mathsf{R}_{\mathsf{CPuf}}(\mathtt{open}) \rangle$ and $\langle \mathsf{C}_{\mathsf{CPuf}}(\mathtt{open}, st_R || p_R), \mathsf{R}_{\mathsf{CPuf}}(\mathtt{open}) \rangle$.
2. Committer $\mathsf{C}_{\mathsf{ExtPuf}}$: send PUF $\mathcal{P}_R$ back to $\mathsf{R}_{\mathsf{ExtPuf}}$.
3. Receiver $\mathsf{R}_{\mathsf{ExtPuf}}$: If both decommitments are successfully completed, then $\mathsf{R}_{\mathsf{ExtPuf}}$ gets the bit b' along with the opening st'_S for ComBit and string $st'_R || p'_R$ for ComResp.
   Check validity of st'_R: query $\mathcal{P}_R$ with $\mathsf{Encode}(st'_S)$ and obtain σ'_R. Compute $st''_R \leftarrow \mathsf{FuzRep}^1(\sigma'_R, p'_R)$. If $st''_R = st'_R$, then accept and output b. Else, reject.

Fig. 2. ExtPuf: Ideal **Extractable** Commitment in the $\mathcal{F}_{\mathsf{PUF}}$ model.

phase, the committer sends x, y, b. The receiver queries the token with x and obtains a string y'. If $y = y'$ the receiver accepts iff $\mathbf{c} = y' \oplus (r \wedge b)$.

The statistically binding property follows from the same arguments of Naor's scheme. The token is sent away before committer can see r. Thus, since x is only n bits, information theoretically the committer cannot instruct a malicious token to output y' adaptively on x. Thus, for any malicious possibly *stateful* token, binding is preserved. The statistically hiding property holds due to the fact that x is secret. A malicious receiver can query the token with any polynomial number of values x'. But, whp she will miss x, and thus she will obtain always 0.

The above protocol is denoted by CTok and is formally described in Fig. 3. We stress that, this is the first construction of unconditional commitment scheme that is secure even against malicious *stateful* tokens.

From Bit Commitment to String Commitment. To commit to a ℓ-bit string using one stateless token only is sufficient to embed ℓ pairs $(x_1, y_1), \ldots, (x_\ell, y_\ell)$ in the

token $\mathcal{T}_{\mathsf{C}}$ and to require that for each i, $x_i \in \{0,1\}^n$ and $y_i \in \{0,1\}^{3\ell n}$. Namely, $\mathcal{T}_{\mathsf{C}}$ grows linearly with the size of the string to be committed. Then, execute protocol CTok for each bit of the string in parallel. The receiver accepts the string iff all bit commitments are accepting.

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

Commitmen Phase

1. Committer $\mathsf{C}_{\mathsf{CTok}}$: pick $x \xleftarrow{\$} \{0,1\}^n$, $y \xleftarrow{\$} \{0,1\}^{3n}$. Create token $\mathcal{T}_{\mathsf{C}}$ implementing the point function $f(x) = y$; $f(x') = 0$ for all $x' \neq x$. Send $\mathcal{T}_{\mathsf{C}}$ to $\mathsf{R}_{\mathsf{CTok}}$.
2. Receiver $\mathsf{R}_{\mathsf{CTok}}$: pick $r \xleftarrow{\$} \{0,1\}^{3n}$. Send r to $\mathsf{C}_{\mathsf{CTok}}$.
3. Committer $\mathsf{C}_{\mathsf{CTok}}$: Send $\mathbf{c} = y \oplus (r \wedge b^{3n})$ to $\mathsf{R}_{\mathsf{CTok}}$.

Decommitment Phase

1. Committer $\mathsf{C}_{\mathsf{CTok}}$: send (b, x) to $\mathsf{R}_{\mathsf{CTok}}$.
2. Receiver $\mathsf{R}_{\mathsf{CTok}}$: query $\mathcal{T}_{\mathsf{C}}$ with x and obtain y. If $b = 0$, check that $\mathbf{c} = y$. Else, check that $y = \mathbf{c} \oplus r$. If the check passes, accept and output b, else reject.

Fig. 3. CTok: Ideal Commitments in the $\mathcal{F}_{\mathsf{wrap}}$ model.

Ideal Extractable Commitment in the $\mathcal{F}_{\mathsf{wrap}}$ model. Extractability is achieved as before. The receiver sends a token $\mathcal{T}_{\mathsf{R}}$ to the committer. The committer is required to query $\mathcal{T}_{\mathsf{R}}$ with the opening of the commitment (namely, the value y) and then commit to the token's response. In the decommitment phase, the committer opens both the commitment of the bit and of the token's response. The receiver then checks that the latter value corresponds to the response of $\mathcal{T}_{\mathsf{R}}$ on input the opening of the commitment of the bit. Note that here the receiver can check the validity of the token's response without physically possessing the token.

We now need to specify the function computed by token $\mathcal{T}_{\mathsf{R}}$. Such function must be resilient against an unbounded adversary that can query the stateless token an arbitrary polynomial number of times.

The function, parameterized by two independent MAC keys $k_{\mathsf{rec}}, k_{\mathsf{tok}}$, takes as input a commitment's transcript $(r, \mathbf{c})$, a MAC-tag σ_{rec} and an opening y. The function checks that y is a valid opening of $(r, \mathbf{c})$, and that σ_{rec} is a valid MAC-tag computed on $(r, \mathbf{c})$ with secret key k_{rec} (i.e., $\sigma_{\mathsf{rec}} = \mathsf{Mac}(k_{\mathsf{rec}}, r||\mathbf{c})$). If both checks are successful, the function outputs the MAC-tag computed on the opening y (i.e., $\sigma_{\mathsf{tok}} = \mathsf{Mac}(k_{\mathsf{tok}}, y)$). Due to the unforgeability of the MAC, and the statistically binding property of the commitment scheme CTok, a malicious committer can successfully obtain the answer to exactly one query. Note that, a malicious committer can perform the following attack. Once it receives the string r from the receiver, it picks strings y_0 and y_1 such that $r = y_0 \oplus y_1$ and sends the commitment $\mathbf{c} = y_0$ to the receiver, obtaining the MAC of $\mathbf{c}$. With the

commitment so computed and the tag, it can query token $\mathcal{T}_{\mathsf{R}}$ twice with each valid opening. In this case, the committer can extract the MAC key, and at the same time baffling the extractor that observes two valid openings. The observation here is that, due to the binding of CTok, for a commitment $\mathbf{c}$ computed in such a way, the malicious committer will not be able, in the decommitment phase, to provide a valid opening. (The reason is that whp she cannot instruct its token to output neither y_0 or y_1). Thus, although the extractor fails and outputs $\perp$, the decommitment will not be accepting. Thus extractability is not violated.

As final step, the committer commits to the token's response σ_{tok}, using the scheme CTok. (If the token of the receiver aborts, the committer sets σ_{tok} to the zero string). In the decommitment phase, the receiver first checks the validity of both commitments (commitment of the bit, commitment of the answer σ_{tok}). Then, given the opening of the bit, it checks that σ_{tok} is a valid MAC computed under key k_{tok} on such opening.

Binding follows from the binding of CTok and the unforgeability of MAC. Hiding still follows from the hiding of CTok. Indeed, the answer of $\mathcal{T}_{\mathsf{R}}$ sent by the malicious receiver, is not forwarded to the receiver, but is committed using the ideal commitment CTok. Furthermore, if $\mathcal{T}_{\mathsf{R}}$ selectivly aborts, the committer does not halt but it continues committing to the zero-string. The receiver will get its token's answer in clear only in the decommitment phase when the bit has been already revealed. The formal description of the above protocol, that we denote by ExtTok, is shown in Fig. 4.

Theorem 3. *Protocol* ExtTok *is an ideal* extractable *commitment in the* $\mathcal{F}_{\mathsf{wrap}}$ *model.*

The proof of Theorem 3 is provided in the full version [18].

Acknowledgments

The second author thanks Dominik Scheder for very interesting discussions on Azuma's inequality, and Akshay Wadia for suggesting a simplification in the compiler. The same author thanks Ivan Visconti and Rafail Ostrovsky for valuable discussions.

References

1. Armknecht, F., Maes, R., Sadeghi, A.R., Standaert, F.X., Wachsmann, C.: A formalization of the security features of physical functions. In: IEEE Symposium on Security and Privacy. pp. 397–412. IEEE Computer Society (2011)
2. Armknecht, F., Maes, R., Sadeghi, A.R., Sunar, B., Tuyls, P.: Memory leakage-resilient encryption based on physically unclonable functions. In: Matsui, M. (ed.) ASIACRYPT. Lecture Notes in Computer Science, vol. 5912, pp. 685–702. Springer (2009)
3. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: Foundations of Computer Science (FOCS'04). pp. 394–403 (2004)

Protocol ExtTok

$(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ is a one-time unconditional MAC. Protocol $\mathsf{CTok} = (\mathsf{C}_{\mathsf{CTok}}, \mathsf{R}_{\mathsf{CTok}})$ is run as sub-routine. Committer's Input: $b \in \{0,1\}$.

Commitment Phase

1. Receiver $\mathsf{R}_{\mathsf{ExtTok}}$: pick MAC-keys: $k_{\mathsf{rec}}, k_{\mathsf{tok}}$. Create token $\mathcal{T}_{\mathsf{R}}$ implementing the following functionality. On input a tuple $(r||\mathbf{c}, \sigma_{\mathsf{rec}}, y)$: if $\mathsf{Vrfy}(k_{\mathsf{rec}}, r||\mathbf{c}, \sigma_{\mathsf{rec}}) = 1$ and ($\mathbf{c} = y$ OR $\mathbf{c} = y \oplus r$) then output $\sigma_{\mathsf{tok}} = \mathsf{Mac}(k_{\mathsf{tok}}, y)$ else output $\perp$. Send $\mathcal{T}_{\mathsf{R}}$ to the sender $\mathsf{C}_{\mathsf{ExtTok}}$.
   Commitment of the Secret Bit: ComBit**.**
2. $\mathsf{C}_{\mathsf{ExtTok}} \Leftrightarrow \mathsf{R}_{\mathsf{ExtTok}}$: run $\langle \mathsf{C}_{\mathsf{CTok}}(\mathtt{com}, b), \mathsf{R}_{\mathsf{CTok}}(\mathtt{com}) \rangle$ so that $\mathsf{C}_{\mathsf{CTok}}$ commits to bit b. Let $(r, \mathbf{c})$ be the transcript of such commitment phase. Let y be the opening of $\mathbf{c}$.
3. Receiver $\mathsf{R}_{\mathsf{ExtTok}}$: compute $\sigma_{\mathsf{rec}} \leftarrow \mathsf{Mac}(k_{\mathsf{rec}}, r||\mathbf{c})$. Send σ_{rec} to Committer $\mathsf{C}_{\mathsf{ExtTok}}$.
4. Committer $\mathsf{C}_{\mathsf{ExtTok}}$: query $\mathcal{T}_{\mathsf{R}}$ with $q = (r||\mathbf{c}, \sigma_{\mathsf{rec}}, y)$ and obtain σ_{tok}. If token $\mathcal{T}_{\mathsf{R}}$ aborts, set $\sigma_{\mathsf{tok}} = 0^n$.
   Commitment of $\mathcal{T}_{\mathsf{R}}$'s Response: ComResp**.**
   $\mathsf{C}_{\mathsf{ExtTok}} \Leftrightarrow \mathsf{R}_{\mathsf{ExtTok}}$: run $\langle \mathsf{C}_{\mathsf{CTok}}(\mathtt{com}, \sigma_{\mathsf{tok}}), \mathsf{R}_{\mathsf{CTok}}(\mathtt{com}) \rangle$ so that $\mathsf{C}_{\mathsf{ExtTok}}$ commits to the response σ_{tok} received from $\mathcal{T}_{\mathsf{R}}$.

Decommitment Phase

1. $\mathsf{C}_{\mathsf{ExtTok}} \Leftrightarrow \mathsf{R}_{\mathsf{ExtTok}}$: opening of both commitments.
   Run $\langle \mathsf{C}_{\mathsf{CTok}}(\mathtt{open}, b), \mathsf{R}_{\mathsf{CTok}}(\mathtt{open}) \rangle$ and $\langle \mathsf{C}_{\mathsf{CTok}}(\mathtt{open}, \sigma_{\mathsf{rec}}), \mathsf{R}_{\mathsf{CTok}}(\mathtt{open}) \rangle$.
2. Receiver $\mathsf{R}_{\mathsf{ExtTok}}$: If both decommitment are successfully completed, then $\mathsf{R}_{\mathsf{ExtTok}}$ gets the bit b' along with the opening y' for ComBit and string σ'_{tok} for ComResp. If $\mathsf{Vrfy}(k_{\mathsf{tok}}, r||y', \sigma'_{\mathsf{tok}}) = 1$ then $\mathsf{R}_{\mathsf{ExtTok}}$ accept and output b'. Else, reject.

Fig. 4. ExtTok: Ideal **Extractable** Commitment in the $\mathcal{F}_{\mathsf{wrap}}$ model.

4. BenOr, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC. pp. 1–10 (1988)
5. Boit, C., Helfmeier, C., Nedospasaov, D., Seifert, J.P.: Cloning physically unclonable functions. IEEE HOST (2013)
6. Brassard, G., Chaum, D., Crépeau, C.: Minimum disclosure proofs of knowledge. J. Comput. Syst. Sci. 37(2), 156–189 (1988)
7. Brzuska, C., Fischlin, M., Schröder, H., Katzenbeisser, S.: Physically uncloneable functions in the universal composition framework. In: Rogaway, P. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 6841, pp. 51–70. Springer (2011)
8. Brzuska, C., Fischlin, M., Schröder, H., Katzenbeisser, S.: Physically uncloneable functions in the universal composition framework. IACR Cryptology ePrint Archive 2011, 681 (2011)
9. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Foundations of Computer Science (FOCS'01). pp. 136–145 (2001)
10. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: TCC. pp. 61–85 (2007)

11. Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. In: Biham, E. (ed.) Advances in Cryptology – EUROCRYPT 2003. Lecture Notes in Computer Science, vol. 2656, pp. 68–86. Springer, Berlin, Germany, Warsaw, Poland (May 4–8, 2003)
12. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai., A.: Universally composable two-party and multi-party secure computation. In: 34th Annual ACM Symposium on Theory of Computing. pp. 494–503. Lecture Notes in Computer Science, ACM Press, Montréal, Québec, Canada (May 19–21, 2002)
13. Chandran, N., Goyal, V., Sahai, A.: New constructions for UC secure computation using tamper-proof hardware. In: Smart, N.P. (ed.) Advances in Cryptology – EUROCRYPT 2008. Lecture Notes in Computer Science, vol. 4965, pp. 545–562. Springer, Berlin, Germany, Istanbul, Turkey (2008)
14. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: STOC. pp. 11–19 (1988)
15. Choi, S.G., Katz, J., Schröder, D., Yerukhimovich, A., Zhou, H.S.: (efficient) universally composable two-party computation using a minimal number of stateless tokens. IACR Cryptology ePrint Archive 2011, 689 (2011)
16. Crépeau, C., Kilian, J.: Achieving oblivious transfer using weakened security assumptions (extended abstract). In: FOCS. pp. 42–52. IEEE Computer Society (1988)
17. Damgård, I., Kilian, J., Salvail, L.: On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In: EUROCRYPT. pp. 56–73 (1999)
18. Damgård, I., Scafuro, A.: Unconditionally secure and universally composable commitments from physical assumptions. IACR Cryptology ePrint Archive 2013, 108 (2013)
19. van Dijk, M., Rührmair, U.: Physical unclonable functions in cryptographic protocols: Security proofs and impossibility results. IACR Cryptology ePrint Archive 2012, 228 (2012)
20. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM J. Comput. 38(1), 97–139 (2008)
21. Döttling, N., Kraschewski, D., Müller-Quade, J.: Unconditional and composable security using a single stateful tamper-proof hardware token. In: Ishai, Y. (ed.) TCC. Lecture Notes in Computer Science, vol. 6597, pp. 164–181. Springer (2011)
22. Döttling, N., Kraschewski, D., Müller-Quade, J.: David & goliath oblivious affine function evaluation - asymptotically optimal building blocks for universally composable two-party computation from a single untrusted stateful tamper-proof hardware token. IACR Cryptology ePrint Archive 2012, 135 (2012)
23. Goyal, V., Ishai, Y., Mahmoody, M., Sahai, A.: Interactive locking, zero-knowledge pcps, and unconditional cryptography. In: Advances in Cryptology – CRYPTO 2010. pp. 173–190. Lecture Notes in Computer Science, Springer, Berlin, Germany, Santa Barbara, CA, USA (Aug 2010)
24. Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding cryptography on tamper-proof hardware tokens. In: Micciancio, D. (ed.) TCC 2010: 7th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 5978, pp. 308–326. Springer, Berlin, Germany, Zurich, Switzerland (Feb 9–11, 2010)
25. Hofheinz, D.: Possibility and impossibility results for selective decommitments. J. Cryptology 24(3), 470–516 (2011)

26. Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: Naor, M. (ed.) Advances in Cryptology – EUROCRYPT 2007. Lecture Notes in Computer Science, vol. 4515, pp. 115–128. Barcelona, Spain (May 20–24, 2007)
27. Katzenbeisser, S., Ko**c**cabas, Ü., Rozic, V., Sadeghi, A.R., Verbauwhede, I., Wachsmann, C.: Pufs: Myth, fact or busted? a security evaluation of physically unclonable functions (pufs) cast in silicon. In: CHES. pp. 283–301 (2012)
28. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing. pp. 723–732. STOC '92, ACM, New York, NY, USA (1992), http://doi.acm.org/10.1145/129712.129782
29. Kolesnikov, V.: Truly efficient string oblivious transfer using resettable tamper-proof tokens. In: Micciancio, D. (ed.) TCC 2010: 7th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 5978, pp. 327–342. Springer, Berlin, Germany, Zurich, Switzerland (Feb 9–11, 2010)
30. Maes, R., Verbauwhede, I.: Physically unclonable functions: A study on the state of the art and future research directions. In: Sadeghi, A.R., Naccache, D. (eds.) Towards Hardware-Intrinsic Security, pp. 3–37. Information Security and Cryptography, Springer Berlin Heidelberg (2010)
31. Moran, T., Segev, G.: David and Goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In: Smart, N.P. (ed.) Advances in Cryptology – EUROCRYPT 2008. Lecture Notes in Computer Science, vol. 4965, pp. 527–544. Springer, Berlin, Germany, Istanbul, Turkey (Apr 13–17, 2008)
32. Müller-Quade, J., Unruh, D.: Long-term security and universal composability. In: TCC. pp. 41–60 (2007)
33. Müller-Quade, J., Unruh, D.: Long-term security and universal composability. J. Cryptology 23(4), 594–671 (2010)
34. Naor, M.: Bit commitment using pseudo-randomness. In: CRYPTO. pp. 128–136 (1989)
35. Ostrovsky, R., Scafuro, A., Visconti, I., Wadia, A.: Universally composable secure computation with (malicious) physically uncloneable functions. IACR Cryptology ePrint Archive 2012, 143 (2012)
36. Ostrovsky, R., Scafuro, A., Visconti, I., Wadia, A.: Universally composable secure computation with (malicious) physically uncloneable functions. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT. Lecture Notes in Computer Science, vol. 7881, pp. 702–718. Springer (2013)
37. Pappu, R.S., Recht, B., Taylor, J., Gershenfeld, N.: Physical one-way functions. Science 297, 2026–2030 (2002)
38. Pappu, R.S.: Physical One-Way Functions. Ph.D. thesis, MIT (2001)
39. Pass, R., Wee, H.: Black-box constructions of two-party protocols from one-way functions. In: Reingold, O. (ed.) TCC. Lecture Notes in Computer Science, vol. 5444, pp. 403–418. Springer (2009)
40. Rührmair, U.: Oblivious transfer based on physical unclonable functions. In: Acquisti, A., Smith, S.W., Sadeghi, A.R. (eds.) TRUST. Lecture Notes in Computer Science, vol. 6101, pp. 430–440. Springer (2010)
41. Rührmair, U., van Dijk, M.: Pufs in security protocols: Attack models and security evaluations. In: IEEE Symposium on Security and Privacy (2013)
42. Sadeghi, A.R., Visconti, I., Wachsmann, C.: Enhancing rfid security and privacy by physically unclonable functions. In: Sadeghi, A.R., Naccache, D. (eds.) Towards Hardware-Intrinsic Security, pp. 281–305. Information Security and Cryptography, Springer Berlin Heidelberg (2010)