Everlasting UC Commitments from Fully Malicious PUFs

Bernardo Magri[1], Giulio Malavolta[2], Dominique Schröder[3], and Dominique Unruh[4]

[1] *The University of Manchester, UK*
[2] *Max Planck Institute for Security and Privacy, Germany*
[3] *Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany*
[4] *University of Tartu, Estonia*

June 7, 2022

Abstract

Everlasting security models the setting where hardness assumptions hold during the execution of a protocol but may get broken in the future. Due to the strength of this adversarial model, achieving any meaningful security guarantees for composable protocols is impossible without relying on hardware assumptions (Müller-Quade and Unruh, JoC'10). For this reason, a rich line of research has tried to leverage physical assumptions to construct well-known everlasting cryptographic primitives, such as commitment schemes. The only known everlastingly UC secure commitment scheme, due to Müller-Quade and Unruh (JoC'10), assumes honestly generated hardware tokens. The authors leave the possibility of constructing everlastingly UC secure commitments from malicious hardware tokens as an open problem. Goyal et al. (Crypto'10) constructs unconditionally UC-secure commitments and secure computation from malicious hardware tokens, with the caveat that the honest tokens must encapsulate other tokens. This extra restriction rules out interesting classes of hardware tokens, such as physically uncloneable functions (PUFs).

In this work we present the first construction of an everlastingly UC-secure commitment scheme in the fully malicious token model *without requiring* honest token encapsulation. Our scheme assumes the existence of PUFs and is secure in the common reference string model. We also show that our results are tight by giving an impossibility proof for everlasting UC-secure *computation* from non-erasable tokens (such as PUFs), even with trusted setup.

1 Introduction

The security of almost all cryptographic schemes relies on certain hardness assumptions. These assumptions are believed to hold right now, and researchers are even fairly certain that they will not be broken in the near future. It is widely believed, for example, that the computational Diffie-Hellmann and the RSA assumptions hold in certain groups. But what about the security of these assumptions in 10, 20, or 100 years? Can we give any formal security guarantees for current constructions that remain valid in the distant future? This is certainly possible for *information-theoretic* schemes

and properties. However, given that many interesting functionalities are *impossible* to realise in an information theoretic sense, this leaves us in a very unsatisfactory situation.

To overcome this problem, Müller-Quade and Unruh suggested a novel security notion widely known as *everlasting universal composability security* [MQU07] (building on the work of Rabin on virtual satellites [Rab03]). The basic idea of this security notion is to bound the running time of the attacker *only* during the protocol execution. After the protocol run is over, the attacker may run in super-polynomial time. This models the intuition that computational assumptions are believed to hold right now, and therefore, during the protocol run. However, at some point in the future, known computational assumptions may no longer hold. Everlasting UC security[1] refers to a composable protocol that remains secure in these settings. The everlasting UC security model has also been considered for quantum protocols [Unr13]. Everlasting UC security is clearly a very desirable security notion, and since it is strictly weaker than statistical UC security, one may hope that it is easier to achieve. However, Müller-Quade and Unruh showed that everlasting UC commitments *cannot* be realised, not even in the common reference string (CRS), or with a public-key infrastructure (PKI) [MQU10].

Everlasting UC Security From Hardware Assumptions. The stark impossibility result of Müller-Quade and Unruh raises the question whether the notion is achievable at all. The authors answered this question affirmatively by presenting two constructions based on hardware assumptions. The first construction is based on a tailored-made hardware token that embeds a random oracle. The second construction relies on signature cards [MQU10]. However, both constructions assume that the hardware token is *honestly* generated. The authors left open the question whether it is possible to achieve everlasting security in the setting of maliciously generated hardware tokens. Goyal et al. [GIMS10] constructs unconditionally UC-secure commitments and secure computation (as opposed to everlasting) from malicious hardware tokens. However, the construction of [GIMS10] requires honest tokens to encapsulate other tokens, ruling out some classes of hardware tokens such as physically uncloneable functions (PUFs).

Physically Uncloneable Functions (PUFs). In this work, we present an everlastingly UC secure commitment scheme assuming the existence of PUFs. Loosely speaking, PUFs are physical objects that can be queried by mapping an input to a specific stimulus and mapping an observable behavior to an output set. The crucial properties for a PUF are (i) that it should be hard (if not impossible) to clone and (ii) that it should be hard to predict the output on any input without first querying the PUF on a close enough input.

1.1 Our Contributions

We initiate the study of everlasting UC security in the setting of maliciously-generated hardware tokens, such as PUFs. Our model extends the frameworks of [BKOV17, CGS08] by introducing *fully* malicious hardware tokens, whose state is not a-priori bounded, the generator of a token can install arbitrary code inside of it, and it can encapsulate (and decapsulate) other (possibly fully malicious) tokens within itself. Our contributions can be summarized as follows:

- Aiming at bridging the gap between hardware tokens and PUFs, we propose a unified ideal functionality for fully malicious tokens that is general enough to capture hardware devices

[1] We use the terms "everlasting security" and "everlasting UC security" interchangeably in this paper to describe protocols that are everlasting secure *and* exhibit a composition theorem which allows a modular design of such protocols.

with arbitrary functionalities such as PUFs and signature cards.

- We put forward a novel definition for unpredictability of PUFs. We argue that the formalization from prior works [OSVW13, BFSK11, Mae13] is not sufficient for our setting because it does not exclude adversaries that may indeed predict the PUF responses for values never queried to the PUF. We demonstrate this fact in Section 4.1.1 by giving a concrete counterexample.
- We show with an impossibility result that one cannot hope to achieve an everlastingly secure oblivious transfer (OT) (therefore, secure computation) in the malicious token setting by using non-erasable (honestly generated) tokens; non-erasable tokens can keep a state but are not allowed to erase previous states.
- Finally, we present an everlastingly UC secure commitment scheme in the fully malicious token model. Our protocol assumes the existence of PUFs and allows for the PUF to be reused for polynomially many runs of the protocol. Our cryptographic building blocks can be instantiated from standard computational assumptions, such as the learning with errors (LWE) problem.

1.2 Related Work

Everlasting and Memory Bound Adversaries. Everlasting security was first considered in the setting of memory-bounded adversaries [CM97, CCM98], and later extended to the UC setting by Müller-Quade and Unruh [MQU10]. Rabin [Rab03] suggested a construction using distributed servers of randomness, called virtual satellites, to achieve everlasting security. The resulting scheme remains secure if the attacker that accesses the communication between the parties and the distributed servers is polynomially bounded during the key exchange. Dziembowski and Maurer [DM08] showed that protocols in the bounded storage model do not necessarily stay secure when composed with other protocols.

Damgård [Dam00] presented a statistical zero-knowledge protocol secure under concurrent composition. Although counterintuitive, statistical zero-knowledge may lose its everlasting property under composition. This was illustrated in [MQU10] for statistically hiding UC commitments [DN02] which were shown to leak secrets under (even sequential) composition; they are composable and statistically hiding, but not at the same time (i.e., the composability only holds for the computational hiding property, intuitively). Technically, the reason for this is that the common reference string used by the simulator is not statistically indistinguishable. For the same reason, the protocol of Damgård [Dam00] does not directly translate into an everlasting commitment scheme: For this specific case, the gap consists in extracting the witness from adversarial proofs using a common reference string that is statistically close to the honestly sampled one.

(Malicious) Hardware Tokens. A model proposed in [Kat07] allows parties to build hardware tokens to compute functions of their choice, such that an adversary, given a hardware token T for a function F, can only observe the input and output behaviour of T. The motivation is that the existence of a tamper-proof hardware can be viewed as a physical assumption, rather than a trust assumption. The authors show how to implement UC-secure two-party computation using stateful tokens, under the DDH assumption. Shortly after, Moran and Segev [MS08] showed that in the hardware token model of [Kat07] even unconditionally secure UC commitments are possible using stateful tokens. This result was later extended by [GIS+10] for unconditionally UC-secure computation, also using stateful tokens.

One limitation of the model of [Kat07] is the assumption that all parties (including the adversary) know the code running inside the hardware token it produces; this assumption gives extra power to the simulator, allowing it to rewind the hardware token in the proofs of [Kat07, MS08, GIS+10]. However, this assumption rules out real scenarios where the adversary can create a new hardware token that simply "encapsulates" a hardware token it receives from some party and that the adversary does not know the code running inside of it.

In this direction, Chandran et al. [CGS08] extended the model of [Kat07] to allow for the hardware tokens produced by the adversary to be stateful, to encapsulate other tokens inside of it and to be passed on to other parties. They constructed a computationally secure UC commitment protocol without setup, assuming the existence of stateless hardware tokens (signature cards). Unfortunately, the construction of [CGS08] cannot fulfil the notion of unconditional (or everlasting) security since it requires perfectly binding, and therefore only computationally hiding, commitments as a building block.

Goyal et al. [GIMS10], following the model of [CGS08] prove that statistically secure OT from stateless tokens is possible if (honest) tokens can encapsulate other tokens. However, honest token encapsulation is highly undesirable in practice, and in particular not even compatible with PUFs as they are physical objects. Interestingly, the authors also show that statistically secure OT (and therefore secure computation) is impossible to achieve when one considers only stateless tokens that cannot be encapsulated. To circumvent this impossibility result, Döttling et al. [DKMQ11, DKMN15] studied the feasibility of secure computation in the stateful token model, where the adversary is not allowed to rewind the token arbitrarily. Although this model has a practical significance, it does not cover certain classes of hardware tokens, such as PUFs. Later, a rich line of research investigated on the round complexity of secure computation using stateless hardware tokens [HPV16, MMQN16] in the computational setting. Unfortunately, it seems that the security guarantees of these protocols cannot be lifted to the everlasting model: In Section 5 we present an impossibility result against everlastingly UC secure computation from stateful but non-erasable honest tokens. The result holds even in the presence of an honestly-sampled CRS.

PUFs. Brzuska et al. [BFSK11] introduced PUFs in the UC framework, and proposed UC constructions of several interesting cryptographic primitives such as oblivious transfer, bit commitment, and key agreement. Ostrovsky, Scafuro, Visconti and Wadia [OSVW13], pointed out that the previous results implicitly assume that all PUFs, including those created by the attacker, are honestly generated. To address this limitation, they defined a model in which an attacker can create malicious PUFs having arbitrary behaviour. Many of the previous protocols can be easily attacked in this new adversarial setting, but Ostrovsky, Scafuro, Visconti and Wadia showed that it is possible to construct universally composable protocols for secure computation in the malicious PUF model under additional, number-theoretic assumptions. They leave open the question of whether unconditional security is possible in the malicious PUF model. Damgård and Scafuro [DS13] have made partial progress on this question presenting a commitment scheme with unconditional security based on PUFs. However, as shown by [BKOV17] in the form of an attack, the construction of [DS13] completely breaks when the the adversary is allowed to generate encapsulated PUFs. Dachman-Soled, Fleischhacker, Katz, Lysyanskaya and Schröder [DFK+14] investigated the possibility of secure two-party computation based on malicious PUFs. Badrinarayanan, Khurana, Ostrovsky, and Visconti [BKOV17] introduced a model where the adversary is allowed to generate malicious PUFs that encapsulate other PUFs inside of it; the outer PUF has oracle access to all its inner PUFs. The security of their scheme assumes a bound on the memory of adversarially generated PUFs.

Reference	Functionality	Model	Honest token	Fully malicious?
Katz [Kat07]	2PC	computational	stateful	✗
Moran and Segev [MS08]	commitments	unconditional	stateful	✗
Chandran et al. [CGS08]	commitments	computational	signature card	✓
Ostrovsky et al. [OSVW13]	2PC	computational	PUF	✗
Damgård and Scafuro [DS13]	commitments	unconditional	PUF	✗
Goyal et al. [GIS+10]	2PC	computational	stateless	✗
Goyal et al. [GIS+10]	2PC	unconditional	stateful	✗
Goyal et al. [GIMS10]	2PC	unconditional	stateless (with token encapsulation)	✓
Dachman-Soled et al. [DFK+14]	2PC	unconditional	PUF	✗
Badrinarayanan et al. [BKOV17]	2PC	unconditional	PUF	✗
Our scheme (section 6)	commitments	everlasting	PUF	✓

Table 1: Comparison of UC secure schemes based on tamper-proof hardware tokens. Fully malicious tokens are the ones whose state is not a-priori bounded, the creator of the token can install arbitrary code inside of it, and the token can encapsulate other (possibly fully malicious) tokens within itself.

In Table 1 we show a comparison of UC schemes based on malicious hardware tokens (including PUFs).

1.3 Technical Overview

In the following we give an informal overview of our everlasting UC commitment scheme construction, and we introduce the main ideas behind our proof strategy. Besides PUFs, our protocol assumes the existence of the following cryptographic building blocks:

- A non-interactive statistically hiding (NI-SH) UC-secure commitment (Com).
- A 2-round statistically receiver private UC-secure oblivious transfer (OT).
- A statistical witness-indistinguishable argument of knowledge (SWIAoK).
- A strong randomness extractor H.

The message flow of our protocol is shown in Figure 1. The protocol is executed by a committer (Alice) and a recipient (Bob). We assume that both parties have access to a uniformly sampled common reference string that contains a random image of a one-way permutation $y = f(x)$.

Protocol Overview. At the beginning of a commitment execution, Bob prepares a series of random string-pairs (p_i^0, p_i^1), and queries them to the PUF to obtain the corresponding pair (q_i^0, q_i^1); the PUF is then transferred to Alice. Here we make the simplifying assumption that a PUF is used only for a single run of the commitment. Note however that one can re-use the same PUF by having Bob computing as many tuples (p_i^0, p_i^1) as needed, and by querying the PUF on all of these values before passing it to Alice.

Alice samples a random string $k \in \{0,1\}^{\ell(\lambda)}$ and engages Bob in many parallel OT instances, where Alice receives $p_i^{k_i}$, and where k_i denotes the i-th bit of k. Alice queries the strings $p_i^{k_i}$ to the PUF and sends to Bob:

- a set of NI-SH commitments $(\mathsf{com}_1, \ldots, \mathsf{com}_{\ell(\lambda)})$ to the outputs of the PUF,
- an (NI-SH) commitment com to m, and
- the string $\omega := H(\mathsf{seed}, k) \oplus m \| \mathsf{decom}$.

Alice then produces a SWIAoK that certifies that either (i) all of her messages were honestly generated, or (ii) she knows a pre-image x such that $f(x) = y$.

The idea here is that, if an algorithm recovers k, then it can also recompute $H(\mathsf{seed}, k)$ and

extract the message m. Note that the value of k is "encoded" in the OT bits of Alice for the $p_i^{k_i}$, and those values are queried by Alice to the PUF. Therefore, an extractor that sees the queries of Alice can easily recover the message m. What is not clear at this point is how to enforce Alice to query the PUF on the correct $p_i^{k_i}$ and not on some other random string. For this reason, we introduce an additional authentication step where Bob publishes all the pairs (q_i^0, q_i^1). In the opening phase, Alice proves (with a SWIAoK) to Bob that the vector of commitments sent in the previous interaction opens indeed to $q_1^{k_1}, \ldots, q_{\ell(\lambda)}^{k_{\ell(\lambda)}}$, up to small errors (or she knows the pre-image of y). Intuitively, Alice cannot convince Bob without querying all the $p_i^{k_i}$, since she would need to guess some $q_i^{\overline{k}_i}$ without knowing the pre-image $p_i^{\overline{k}_i}$ (due to the security of the OT). In the proof, the extractor can recover k by just looking at the queries Alice made to the PUF.

To see why the commitment is hiding, it is sufficient to observe that k hides the message in an information theoretic sense, under the assumption that the OT and SWIAoK protocols are secure. One subtlety that we need to address is that some bits of k might be revealed by the aborts of Alice. For this reason, we one-time-pad the message m with $H(\mathsf{seed}, k)$: The strong randomness extractor guarantees that the value $H(\mathsf{seed}, k)$ is still uniformly distributed even if some bits of k are leaked.

Proof Sketch (Hiding). We show that our commitment scheme is hiding through a series of hybrids where at the last step Alice can equivocate the commitment to any message of her choice. Note that every step is information-theoretic.

$\mathcal{H}_1$: Alice uses x, the pre-image of y, as a witness to compute the SWIAoK. Since the AoK is statistically witness indistinguishable, this hybrid is statistically close to the original protocol.

$\mathcal{H}_2$: Alice uses the simulator for the OT protocols and extracts both values (p_i^0, p_i^1). Since the OT is statistically receiver-private, this hybrid is statistically close to the previous. In the full proof this is shown via a hybrid argument.

$\mathcal{H}_3$: Alice computes com_i as commitment to a random string. A hybrid argument can be used to bound the distance of this hybrid with the previous by the statistically-hiding property of the commitment scheme.

$\mathcal{H}_4$: Alice chooses the value of k for all sessions upfront. Here the change is only syntactical.

$\mathcal{H}_5$: Alice no longer queries the PUF token but instead checks that the output pairs (q_i^0, q_i^1) sent by Bob correspond to the correct outputs of the PUF on input (p_i^0, p_i^1). Note that the state of the PUF is fixed once the PUF is sent to Alice and therefore the consistency of all pairs (q_i^0, q_i^1) is well defined. Note that the relation is not efficiently computable by Alice, but for information-theoretic security the fact that it is defined is enough. Since Alice retains the ownership of the PUF, this hybrid is identical to the previous.

$\mathcal{H}_5$: Alice samples ω uniformly at random. Note that in $\mathcal{H}_4$ the leakage of Alice of k is bounded by whether she aborts or not. Since Alice aborts at most once and since there are at most polynomially-many sessions, we can bound the leakage of k to $O(\log \lambda)$ bits. Leveraging the randomness extractor H, we can argue that $\mathcal{H}_4$ and $\mathcal{H}_5$ are statistically indistinguishable.

$\mathcal{H}_6$: Alice opens the commitment to a message of her choice. Note that in $\mathcal{H}_5$ the original message m is information-theoretically hidden.

Proof Sketch (Binding). To argue that the scheme is binding we define the following extractor: The algorithm examines the list of queries made by Alice to the PUF and, for each i, it checks whether some query q is equal to p_i^b (for $b \in \{0, 1\}$), if this is the case then it sets $k_i = b$. Once

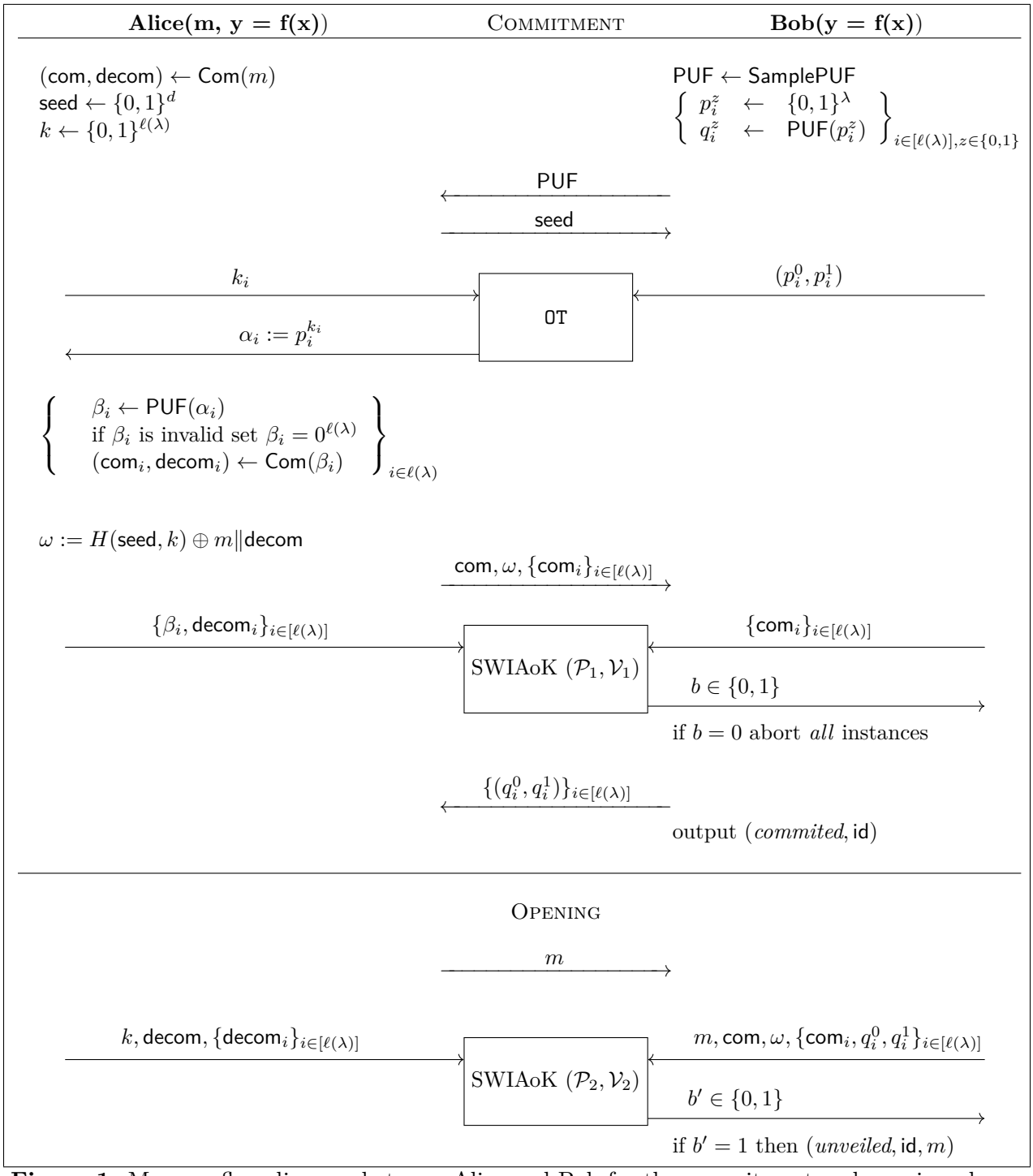

Figure 1: Message flow diagram between Alice and Bob for the commitment and opening phases of our protocol in Section 6.

the full k is reconstructed, the extractor computes $\omega \oplus H(\mathsf{seed}, k) = m \| \mathsf{decom}$ and outputs m. To show that the extractor always succeeds we need to argue that:

1. The value of k is always well-defined: If some $\mathsf{q} = p_i^0$ and some other query $\mathsf{q}' = p_i^1$, then the bit k_i is not well defined. However, this means that Alice learned both p_i^0 and p_i^1 from the OT protocol, which is computationally infeasible.
2. The string k is always fully reconstructed: If no query q is equal to p_i^0 or p_i^1, then the i-th bit of k is not defined. This implies that Alice never queried p_i^0 or p_i^1 to the PUF. However note that Alice should produce a commitment com_i to either $\mathrm{PUF}(p_i^0)$ or $\mathrm{PUF}(p_i^0)$ and prove consistency. This is clearly not possible without querying the PUF unless Alice breaks the binding of the commitment or proves a false statement in the SWIAoK. To establish the latter, we also need to rule out the case where Alice computes the SWIAoK using the knowledge of x, the pre-image of y. In the full proof we show this via a reduction against the one-wayness of the one-way permutation f.

We are now in the position to show that the extracted message m is identical to the one that Alice decommits to. Recall that Alice proves that she committed to the values $\mathrm{PUF}(p_i^{k_i})$ such that $\omega \oplus H(\mathsf{seed}, k) = m \| \mathsf{decom}$. It follows that, if k is uniquely defined, then the extractor always returns the correct m, unless Alice can break the soundness of the SWIAoK (or inverts the one-way permutation). By the above conditions, this happens with all but negligible probability.

On the Common Reference String. Our protocol needs to assume the existence of a common reference string to equivocate commitments in the security proof: Having access to the generation of the crs, the simulator can craft proofs for false statements, simulate the OT, and extract the commitments. Note that the simulation has to be "straight-line", since we cannot rewind the adversary in the UC framework. A previous work [OOR+14] circumvented this issue by leveraging some computationally hard problem. Unfortunately this class of techniques does not seem to apply to the everlasting setting since the environment can distinguish a simulated trace once it becomes unbounded. The work of [DS13] builds unconditionally secure commitments from PUFs without a CRS, but as shown by [BKOV17], the construction breaks down in our model where the the adversary is allowed to generate encapsulated PUFs. It is not clear if the techniques of [DS13] can be adapted to our setting. We leave the question of removing the necessity of a common reference string from our protocol as a fascinating open problem.

2 Preliminaries

In the following we introduce the notation and the building blocks necessary for our results.

2.1 Notations

An algorithm $\mathcal{A}$ is *probabilistic polynomial-time* (PPT) if $\mathcal{A}$ is randomized and for any input $x, r \in \{0,1\}^*$ the computation of $\mathcal{A}(x; r)$ terminates in at most $\mathsf{poly}(|x|)$ steps. We denote with $\lambda \in \mathbb{N}$ the security parameter. A function negl is *negligible*, if for any positive polynomial p and sufficiently large k, $\mathsf{negl}(k) < 1/p(k)$. A relation $R \in \{0,1\}^* \times \{0,1\}^*$ is an $\mathcal{NP}$ relation if there is a polynomial-time algorithm that decides $(x, w) \in R$. If $(x, w) \in R$, then we call x the *statement* and x *witness* for x. We denote by $\mathsf{hd}(x, x')$ the Hamming distance between two bitstrings x and x'. Given two ensembles $\mathbf{X} = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathbf{Y} = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$, we write $\mathbf{X} \approx \mathbf{Y}$ to denote that the two ensembles are statistically indistinguishable, and $\mathbf{X} \approx_c \mathbf{Y}$ to denote that they are computationally

indistinguishable. We denote the set $\{1, \ldots, n\}$ by $[n]$. We recall the definition of statistical distance.

Definition 1 (Statistical Distance)**.** Let X and Y be two random variables over a finite set $\mathcal{U}$. The statistical distance between X and Y is defined as

$$\mathbb{SD}[X, Y] = \frac{1}{2} \sum_{u \in \mathcal{U}} |\Pr[X = u] - \Pr[Y = u]| .$$

2.2 Cryptographic Building Blocks

One Way Function. A one-way function is a function that is easy to compute and hard to invert. It is the building block of almost all known cryptographic primitives.

Definition 2. A function $f : \{0,1\}^* \to \{0,1\}^*$ is one way if and only if it can be computed in polynomial time but for all PPT algorithms $\mathcal{A}$, there exists a negligible function negl such that

$$\Pr\left[x' \leftarrow \mathcal{A}(1^\lambda, f(x)) : f(x') = f(x)\right] \leq \mathsf{negl}(\lambda),$$

where the probability is taken over the random choice of x. Moreover, we say that f is a one-way permutation whenever the domain and range of f are of the same size.

Non-interactive Commitment Scheme. A commitment scheme (in the CRS model) consists of a pair of efficient algorithms $\mathcal{C} = (\mathsf{Com}, \mathsf{Open})$ where: Com takes as input $m \in \{0,1\}^\lambda$ and outputs $(\mathsf{decom}, \mathsf{com}) \leftarrow \mathsf{Com}(m)$, where decom and com are both of length $\{0,1\}^\lambda$; the algorithm $\mathsf{Open}(\mathsf{decom}, \mathsf{com})$ outputs a message m or $\perp$ if c is not a valid commitment to any message. It is assumed that the commitment scheme is complete, i.e., for any message $m \in \{0,1\}^\lambda$ and $(\mathsf{decom}, \mathsf{com}) \leftarrow \mathsf{Com}(ck, m)$, we have $\mathsf{Open}(ck, \mathsf{decom}, \mathsf{Com}(ck, m)) = m$ with overwhelming probability in $\lambda \in \mathbb{N}$. For convenience, we assume that the verification is deterministic and canonical (i.e., it takes as input the random coins used in the commitment phase and checks whether the commitment was correctly computed).

We require commitments to be (stand-alone) statistically hiding. Let $\mathcal{A}$ be a non-uniform adversary against $\mathcal{C}$ and define its hiding-advantage as

$$\mathbf{Adv}^{\mathrm{hid}}_{\mathcal{C},\mathcal{A}}(\lambda) = 2 \cdot \Pr\left[b = b' \;\middle|\; \begin{array}{l} (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}(1^\lambda); b \leftarrow \{0,1\}; \\ (\mathsf{decom}, \mathsf{com}) \leftarrow \mathsf{Com}(m_b); b' \leftarrow \mathcal{A}(\mathsf{com}, \mathsf{st}) \end{array}\right] - 1 .$$

Definition 3 (Statistically Hiding)**.** $\mathcal{C}$ is statistically hiding if the advantage function $\mathbf{Adv}^{\mathrm{hid}}_{\mathcal{C},\mathcal{A}}$ is a negligible function for all unbounded adversaries $\mathcal{A}$.

Furthermore, we require the commitments to be UC-secure: Roughly speaking, an *equivocator* (with the help of a trapdoor in the CRS) can open the commitments arbitrarily. On the other hand, we require the existence of a computationally indistinguishable CRS (in *extraction* mode) where commitments are statistically binding and can be efficiently extracted via the knowledge of a trapdoor. Such commitments can be constructed in the CRS model from a variety of assumptions [PW08], including the learning with errors (LWE) problem. For a precise functionality we refer the reader to Section 3.3.

Oblivious Transfer. A $\binom{2}{1}$-Oblivious transfer (OT) is a protocol executed between two parties called sender $\mathcal{S}$ (i.e. Alice) with input bits (s_0, s_1) and receiver $\mathcal{R}$ (i.e. Bob) with input bit b. Bob wishes to retrieve s_b from Alice in such a way that Alice does not learn anything about Bob's choice b and Bob learns nothing about Alice's remaining input s_{1-b}. In this work, we require a 2-round protocol $(\mathsf{Sender}_{\mathsf{OT}}, \mathsf{Receiver}_{\mathsf{OT}})$ secure in the CRS model, which satisfies (stand-alone) statistical receiver-privacy. We define the sender Alice's advantage of breaking the security of Bob to be

$$\mathbf{Adv}_{\mathcal{S}}^{\mathsf{OT}} = \left| \Pr\left[b \leftarrow \mathcal{A}\left(\mathsf{Receiver}_{\mathsf{OT}}(b)\right)\right] - \frac{1}{2} \right|.$$

Definition 4 (Statistical Receiver Privacy)**.** $(\mathsf{Sender}_{\mathsf{OT}}, \mathsf{Receiver}_{\mathsf{OT}})$ is statistically receiver-private if the advantage function $\mathbf{Adv}_{\mathcal{S}}^{\mathsf{OT}}$ is a negligible function for all unbounded adversaries $\mathcal{A}$.

In addition, we require our OT to be UC-secure: For a well-formed CRS, there exists an efficient equivocator that can (non-interactively) recover both messages of the sender. Furthermore, there exists an alternative CRS distribution (which is computationally indistinguishable from the original one) and an efficient non-interactive extractor that is able to uniquely recover the message of the receiver using the knowledge of a trapdoor. Such 2-round OT can be constructed from a variety of assumptions [PVW08], including LWE [Qua20]. For a precise description of the ideal functionality, we refer the reader to Section 3.3.

Statistical Witness-Indistinguishable Argument of Knowledge (SWIAoK). A witness-indistinguishable argument is a proof system for languages in $\mathcal{NP}$ that does not leak any information about which witness the prover used, not even to a malicious verifier. If the prover is a PPT algorithm, then we call such a system an argument system, and if it is unbounded, we call it a proof system. For witness-indistinguishable arguments of knowledge we formally introduce the following notation to represent interactive executions between algorithms $\mathcal{P}$ and $\mathcal{V}$. By $\langle \mathcal{P}(y), \mathcal{V}(z) \rangle (x)$ we denote the view (i.e., inputs, internal coin tosses, incoming messages) of $\mathcal{V}$ when interacting with $\mathcal{P}$ on common input x, when $\mathcal{P}$ has auxiliary input y and $\mathcal{V}$ has auxiliary input z. Some of the following definitions are based on [OOR+14].

Definition 5 (Witness Relation)**.** A witness relation for a $\mathcal{NP}$ language L is a binary relation R that is polynomially bounded, polynomial time recognizable, and characterizes L by $L = \{x : \exists w \text{ s.t. } (x, w) \in R\}$. We say that w is a witness for $x \in L$ if $(x, w) \in R$.

Definition 6 (Interactive Argument System)**.** A two-party game $\langle \mathcal{P}, \mathcal{V} \rangle$ is called an Interactive Argument System for a language L if $\mathcal{P}$, $\mathcal{V}$ are PPT algorithms and the following two conditions hold:

- Completeness: For every $x \in L$, $\Pr\left[\langle \mathcal{P}, \mathcal{V} \rangle (x) = 1 \right] = 1$.
- Soundness: For every $x \notin L$ and every PPT algorithm $\mathcal{P}^*$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that, $\Pr\left[\langle \mathcal{P}^*, \mathcal{V} \rangle (x) = 1 \right] \leq \mathsf{negl}(|x|)$.

Definition 7 (Witness Indistinguishability)**.** Let $L \in \mathcal{NP}$ and $(\mathcal{P}, \mathcal{V})$ be an interactive argument system for L with perfect completeness. The proof system $(\mathcal{P}, \mathcal{V})$ is *witness indistinguishable* (WI) if for every PPT algorithm $\mathcal{V}^*$, and every two sequences $\{w_x^1\}_{x \in L}$ and $\{w_x^2\}_{x \in L}$ such that $w_x^1, w_x^2 \in R$, the following sequences are witness indistinguishable:

1. $\{\langle \mathcal{P}(w_x^1), \mathcal{V}(z) \rangle (x)\}_{x \in L, z \in \{0,1\}^*}$
2. $\{\langle \mathcal{P}(w_x^2), \mathcal{V}(z) \rangle (x)\}_{x \in L, z \in \{0,1\}^*}$

Next, we define the notion of extractability for SWIAoKs.

Definition 8 (Argument of Knowledge)**.** Let $L \in \mathcal{NP}$ and $(\mathcal{P}, \mathcal{V})$ be an interactive argument system for L with perfect completeness. The proof system $(\mathcal{P}, \mathcal{V})$ is an *argument of knowledge* (AoK) if there exists a PPT algorithm Ext, called the extractor, a polynomial p, and a constant c such that, for every PPT machine $\mathcal{P}^*$, every $x \in L$, auxiliary input z, and random coins r, there exists a negligible function negl such that

$$\Pr\left[\mathsf{Ext}^{\mathcal{P}^*(z,r)}(x) = w : (x,w) \in R\right] \geq \frac{1}{p} \cdot \Pr[\langle \mathcal{P}^*(z;r), \mathcal{V}(x)\rangle = 1]^c - \mathsf{negl}(\lambda).$$

Strong Randomness Extractor. A strong randomness extractor is a function that, applied to some input with high min-entropy, returns some uniformly distributed element in the range.

Definition 9 (Strong Randomness Extractor)**.** A function $H : \{0,1\}^d \times \{0,1\}^\ell \to \{0,1\}^c$ is called a (t, ε)-strong randomness extractor if for all $X \in \{0,1\}^\ell$ s.t $\mathsf{H}_\infty(X) \geq t$, we have that,

$$\mathbb{SD}\left((U_d, H(U_d, X)), (U_d, U_c)\right) \leq \varepsilon$$

and $L = t - c$ is called the entropy loss of H.

3 Universal Composability Framework

In this section we recall the basics of the Universal Composability (UC) framework of Canetti [Can01], and later we discuss the Everlasting Universal Composability framework[2] following [MQU10] closely. We refer the reader to [Can01, MQU10] for a more comprehensive description.

3.1 Basics of the UC Framework

Our description of the UC framework follows [MQU10] closely. The composition of two provably secure protocols does not necessarily preserve the security of each protocol and the result may also be no longer secure. A framework that analyses the security of composed protocols and which is able to provide security guarantees is the Universal Composability framework (UC) due to Canetti [Can01].

The main idea of this security notion is to compare a real protocol π with some ideal protocol ρ. In most cases, this ideal protocol ρ will consist of a single machine, a so-called ideal functionality. Such a functionality can be seen as a trusted machine that implements the intended behaviour of the protocol. For example, a functionality $\mathcal{F}$ for commitment would expect a value m from a party C. Upon receipt of that value, the recipient R would be notified by $\mathcal{F}$ that C has committed to some value (but $\mathcal{F}$ would not reveal that value). When C sends an unveil request to $\mathcal{F}$, the value m will be sent to R (but $\mathcal{F}$ will not allow C to unveil a different value).

Given a real protocol π and an ideal protocol ρ, we say that π realises ρ (also called "implements", "emulates", or "is as secure as") if for any adversary $\mathcal{A}$ attacking the protocol π there is a simulator $\mathcal{S}$ performing an attack on the ideal protocol ρ such that no environment $\mathcal{Z}$ can distinguish between π running with $\mathcal{A}$ and ρ running with $\mathcal{Z}$. Here $\mathcal{Z}$ may choose the protocol

[2]The framework was called "Long-term UC" in [MQU07] and renamed to "Everlasting UC" in the follow-up work [MQU10].

inputs and read the protocol outputs and may communicate with the adversary or simulator (but $\mathcal{Z}$ is, of course, not informed whether it communicates with the adversary or the simulator). First, the environment may communicate with the adversary during the protocol execution, and second, the environment does not need to choose the inputs at the beginning of the protocol execution; it may adaptively send inputs to the protocol parties at any time, and it may choose these inputs depending upon the outputs and the communication with the adversary. These modifications are the reason for the very strong composability properties of the UC model.

Network Execution. In the UC framework, all protocol machines and functionalities, as well as the adversary, the simulator and the environment are modelled as interactive Turing machines (ITM). Throughout a protocol execution, an integer k called the security parameter is accessible to all parties. At the beginning of the execution of a network consisting of π, $\mathcal{A}$, and $\mathcal{Z}$, the environment $\mathcal{Z}$ is invoked with an initial input z. From then on, every machine M that is activated can send a message m to a single other machine M'. Then that machine M' is activated and given the message m and the id of the originator M'. If in some activation a machine does not send a message, the environment $\mathcal{Z}$ is activated again. Additionally the environment may issue corruption requests for some party P. From then on, the machines corresponding to the party P are controlled by the adversary (i.e., it can send and receive messages in the name of that machine, and it can read the internal state of that machine). Finally, at some point the environment $\mathcal{Z}$ gives some output m which can be an arbitrary string. By $\mathrm{EXC}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)$ we denote the distribution of that output m on security parameter k and initial input z. Analogously, we define $\mathrm{EXC}_{\rho,\mathcal{S},\mathcal{Z}}(k,z)$ for an execution involving the protocol ρ, the simulator $\mathcal{S}$, and the environment $\mathcal{Z}$.

We distinguish two different flavours of corruption. We speak of static corruption if the environment $\mathcal{Z}$ may only send corruption requests before the begin of the protocol, and of adaptive corruption if $\mathcal{Z}$ may send corruption requests at any time in the protocol, even depending on messages learned during the execution. In this paper, we will restrict our attention to the less strict security model using static corruption. We leave the case of adaptive corruptions, in which the environment may corrupt any party adaptively during the execution of the protocol as an interesting open problem.

UC Definitions. If the ideal protocol ρ consists of an ideal functionality $\mathcal{F}$, for technical reasons we assume the presence of so-called dummy parties that forward messages between the environment $\mathcal{Z}$ and the functionality $\mathcal{F}$. For example, assume that $\mathcal{F}$ is a commitment functionality. In an ideal execution, $\mathcal{Z}$ would send a value m to the party C (since it does not know of $\mathcal{F}$ and therefore will not send to $\mathcal{F}$ directly). Then C would forward m to $\mathcal{F}$. Then $\mathcal{F}$ notifies R that a commitment has been performed. This notification is then forwarded to $\mathcal{Z}$. With these dummy parties we have, at least syntactically, the same messages as in the real execution: $\mathcal{Z}$ sends m to C and receives a commit notification from R. Second, the dummy-parties allow a meaningful corruption in the ideal model. If $\mathcal{Z}$ corrupts some party P, in the ideal model the effect would be that the simulator controls the corresponding dummy party P and thus can read and modify messages to and from the functionality $\mathcal{F}$ in the name of P. Thus if we write $\mathrm{EXC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$, this is essentially an abbreviation for $\mathrm{EXC}_{\rho,\mathcal{S},\mathcal{Z}}$ where the ideal protocol ρ consists of the functionality $\mathcal{F}$ and the dummy-parties. Having defined the families of random variables $\mathrm{EXC}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)$ and $\mathrm{EXC}_{\rho,\mathcal{S},\mathcal{Z}}(k,z)$ we can now define security via indistinguishability.

Definition 10 (Universal Composability [Can01])**.** A protocol π *UC realises* a protocol ρ, if for any polynomial-time adversary $\mathcal{A}$ there exists a polynomial-time simulator $\mathcal{S}$, such that for any

polynomial-time environment $\mathcal{Z}$,

$$\{\mathrm{EXC}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^{poly(k)}} \approx_c \{\mathrm{EXC}_{\rho,\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^{poly(k)}}.$$

Note that in this definition, it is also possible to only consider environments $\mathcal{Z}$ that give a single bit of output. As demonstrated in [Can01], this gives rise to an equivalent definition. However, in the case of everlasting UC below, this will not be the case, so we stress the fact that we allow $\mathcal{Z}$ to output arbitrary strings. In particular an environment machine can output its complete view.

Natural variants of this definition are statistical UC, where all machines (environment, adversary, simulator) are computationally unbounded and the families of random variables are required to be statistically indistinguishable, and perfect UC, where all machines are computationally unbounded and the families of random variables are required to have the same distribution. In these cases one often additionally requires that if the adversary is polynomial-time, so is the simulator.

Composition. For some protocol σ, and some protocol π, by σ^π we denote the protocol where σ invokes (up to polynomially many) instances of π.[3] That is, in σ^π the machines from σ and from π run together in one network, and the machines from σ access the inputs and outputs of π. (In particular, $\mathcal{Z}$ then talks only to σ and not to the subprotocol π directly.) A typical situation would be that $\sigma^\mathcal{F}$ is some protocol that makes use of some ideal functionality $\mathcal{F}$ (say, a commitment) and then σ^π would be the protocol resulting from implementing that functionality by some protocol π (say, a commitment protocol). One would hope that such an implementation results in a secure protocol σ^π. That is, if π realises $\mathcal{F}$ and $\sigma^\mathcal{F}$ realises $\mathcal{G}$, then σ^π realises $\mathcal{G}$. Fortunately, this is the case:

Theorem 11 (Universal Composition Theorem [Can01])**.** *Let π, ρ, and σ be polynomial-time protocols. Assume that π UC realises ρ. Then σ^π UC realises σ^ρ.*

The intuitive reason for this theorem is that σ can be considered as an environment for π or ρ, respectively. Since Definition 10 guarantees that π and ρ are indistinguishable by any environment, security follows. In a typical application of this theorem, one would first show that π realises $\mathcal{F}$ and that $\sigma^\mathcal{F}$ realises $\mathcal{G}$. Then using the composition theorem one gets that σ^π realises $\sigma^\mathcal{F}$ which in turn realises $\mathcal{G}$. Since the realises-relation is transitive (as can be easily seen from Definition 10), it follows that σ^π realises $\mathcal{G}$.

This composition theorem is the main feature of the UC framework. It allows us to build up protocols from elementary building blocks. This greatly increases the manageability of security proofs for large protocols. Furthermore it guarantees that the protocol can be used in arbitrary contexts. Analogous theorems also hold for statistical and perfect UC.

Dummy-adversary. When proving the security of a given protocol in the UC setting, a useful tool is the so-called dummy-adversary. The dummy-adversary $\tilde{\mathcal{A}}$ is the adversary that simply forwards messages between the environment $\mathcal{Z}$ and the protocol (i.e., it is a puppet of the environment that does whatever $\mathcal{Z}$ instructs it to do). In [Can01] it is shown that UC security with respect to the dummy-adversary implies UC security. The intuitive reason is that since $\tilde{\mathcal{A}}$ does whatever $\mathcal{Z}$ instructs it to do, it can perform arbitrary attacks and is therefore the worst-case adversary given the right environment (remember that we quantify over all environments).

[3] For simplicity, we assume throughout this work that the session ids assigned to these instances are $\{1,\ldots,p\}$ for some polynomial p.

We very roughly sketch the proof idea. Let protocols π and ρ and some adversary $\mathcal{A}$ be given. Assume that π UC realises ρ with respect to the dummy-adversary $\tilde{\mathcal{A}}$. We want to show that π UC realises ρ with respect to $\mathcal{A}$. Given an environment $\mathcal{Z}$, we construct an environment $\mathcal{Z}_{\mathcal{A}}$ which simulates $\mathcal{Z}$ and $\mathcal{A}$. Note that an execution of $\mathrm{EXC}_{\pi,\tilde{\mathcal{A}},\mathcal{Z}_{\mathcal{A}}}$ is essentially the same as $\mathrm{EXC}_{\pi,\mathcal{A},\mathcal{Z}}$ (up to a regrouping of machines). Then there is a simulator $\tilde{\mathcal{S}}$ such that $\mathrm{EXC}_{\pi,\tilde{\mathcal{A}},\mathcal{Z}_{\mathcal{A}}}$ and $\mathrm{EXC}_{\rho,\tilde{\mathcal{S}},\mathcal{Z}_{\mathcal{A}}}$ are indistinguishable. Let $\mathcal{S}$ be the simulator that internally simulates the machines $\mathcal{A}$ and $\tilde{\mathcal{S}}$ and forwards all actions performed by $\mathcal{A}$ as instructions to $\tilde{\mathcal{S}}$ (remember that $\tilde{\mathcal{S}}$ simulates $\tilde{\mathcal{A}}$, so it expects such instructions). Then $\mathrm{EXC}_{\rho,\tilde{\mathcal{S}},\mathcal{Z}_{\mathcal{A}}}$ is again the same as $\mathrm{EXC}_{\rho,\mathcal{S},\mathcal{Z}}$ up to a regrouping of machines. Summarising, we have that $\mathrm{EXC}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\mathrm{EXC}_{\rho,\mathcal{S},\mathcal{Z}}$ are indistinguishable.

A nice property of this technique is that it is quite robust with respect to changes in the definition of UC security. For example, it also holds with respect to statistical and perfect UC security, as well as with respect to the notion of Everlasting UC from [MQU10].

3.2 Everlasting UC Security

In this section, we present our definitions of everlasting UC security. Our formalization builds on Canetti's Universal Composability framework [Can01] and extends the notion of everlasting/long-term security due to Müller-Quade and Unruh [MQU10]. Loosely speaking, everlasting security guarantees the "standard" notion of UC security during the execution of the protocol. This means that the security is guaranteed against polynomially bounded adversaries. Therefore, standard computational assumptions, such as the hardness of the decisional Diffie-Hellman problem and the existence of one-way functions can be used as hardness assumptions. However, after the execution of the protocol, we no longer assume that these assumptions hold, because they may be broken in the future. Müller-Quade and Unruh model this by letting the distinguisher become unbounded *after the execution* of the protocol. Everlasting security guarantees security and confidentiality in this setting.

They showed in [MQU10] that everlasting UC commitments *cannot* be realised, not even in the common reference string (CRS) or the public-key infrastructure (PKI) model.[4] The fact that everlasting UC commitments cannot be constructed in the CRS model shows a strong separation between the everlasting UC and the computational UC security notion, because commitments schemes do exist (under standard assumptions) in the computational UC security model [CF01]. The stark impossibility result of Müller-Quade and Unruh motivated the use of other trust assumptions, such as trusted pseudorandom functions (TDF) and signature cards [MQU10]. It is not hard to see that everlasting UC security is strictly *stronger* than computational UC security, since the adversary is allowed to become unbounded after the execution of the protocol, and it is strictly *weaker* than statistical UC security, since the adversary is polynomially bounded during the run of the protocol.

3.2.1 Defining Everlasting UC Security.

The formalization of [MQU10] is surprisingly simple and only extends the original UC definition by the requirement that the execution of the real protocol and of the functionality cannot be distinguished by an unbounded entity *after* the execution of the protocol is over (that is run by

[4] Interestingly, UC commitments that are statistically hiding can be constructed in the CRS model, as shown by [DN02]. But [MQU10] later shows that those commitments lose their statistically hiding property under composition.

efficient adversaries and environments). Formally, this means that the output of the environment in the real and ideal worlds is *statistically* close. A comprehensive discussion is given in [MQU10], and we briefly recall the definitions.

Definition 12 (Everlasting UC)**.** A protocol π *everlastingly UC-realizes* an ideal protocol ρ if, for any PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that, for any PPT environment $\mathcal{Z}$,

$$\{EXC_{\pi,\mathcal{A},\mathcal{Z}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}} \approx \{EXC_{\rho,\mathcal{S},\mathcal{Z}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}}.$$

In [MQU10], the authors show that the composition theorem from [Can01] also holds with respect to Definition 12. A shortcoming of Definition 12, when applied to the token model, is that the distinguisher has no access to the hardware token after it becomes unbounded. Another issue is that Definition 12 does not model the case that the hardware assumption may be broken in the long-term.

3.2.2 Everlasting UC Security with Hardware Assumptions.

We define a notion of everlasting security which allows the participants in a protocol to leak information in the long term.

With the exception of the environment $\mathcal{Z}$ and the adversary $\mathcal{A}$, we give each *instance* of a Turing machine (ITI for short) in the protocol an additional output tape, that we call *long-term output tape*. We modify the execution model to handle the long-term tapes as follows. At the end of the execution of the protocol (i.e., when the environment $\mathcal{Z}$ produces its output m), adversary $\mathcal{A}$ is invoked once again, this time with all long-term tapes, and produces an output a. We define the new execution model to be $EXC' := (m, a)$. A formal definition follows.

Definition 13 (Everlasting UC with Long-term Tapes)**.** A protocol π *everlastingly UC realizes* an ideal protocol ρ if, for any PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that, for any PPT environment $\mathcal{Z}$,

$$\{EXC'_{\pi,\mathcal{A},\mathcal{Z}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}} \approx \{EXC'_{\rho,\mathcal{S},\mathcal{Z}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}}$$

In Definition 13, the distinguisher does not get the long-term tapes directly, instead, the tapes go through the adversary. The real adversary $\mathcal{A}$ can, wlog, let the tapes go unchanged to the distinguisher (i.e., dummy-adversary). The simulator $\mathcal{S}$ can replace the long-term tapes by any simulated a of its choice. We point out that Definition 13 is equivalent to Definition 12 when none of the ITIs in π or ρ have long-term output tapes.

It is easy to show that the composition theorem from [MQU10] carries over to our settings: The long-term tapes of the honest parties are also given to the adversary/simulator at the end of the protocol execution, however the simulator (when communicating with the environment) can replace them with values of his choice. Formally, this means that the long-term tapes are just a message sent from protocol to adversary (in the same way as, e.g., the state is sent in the case of adaptive corruption), and consequently when proving the composition theorem, those messages are handled in exactly the same way as the messages resulting from adaptive corruption.

3.3 Functionalities

In this section, we define some commonly used functionalities that we will need for our results.

CRS. The first functionality is the common reference string (CRS). Intuitively, the CRS denotes a string sampled uniformly from a given distribution $\mathcal{G}$ by some trusted party, and that is known to all parties prior to the start of the protocol.

Definition 14 (Common reference string (CRS))**.** Let $\mathcal{D}_\lambda$ ($\lambda \in \mathbb{N}$) be an efficiently samplable distribution on $\{0,1\}^*$. At the beginning of the protocol, the functionality $\mathcal{F}_{\text{CRS}}^{\mathcal{D}}$ chooses a value r according to the distribution $\mathcal{D}_\lambda$ (where λ is the security parameter) and sends r to the adversary and all parties P_i.

Multiple commitment. Here we recall the functionality for a commitment scheme. Throughout the following description we implicitly assume that the attacker is informed about each invocation and that the attacker controls the output of the functionality. We omit those messages from the description of the functionalities for readability. Note that to securely realize this functionality, a protocol must guarantee independence among different executions of the commitment protocol.

Definition 15 (Multiple Commitment)**.** Let S and R be two parties, where we call S the sender and R the receiver. The functionality $\mathcal{F}_{\mathsf{MCOM}}^{S\to R,\ell}$ behaves as follows: Upon the command $(\texttt{commit}, \mathsf{sid}, x)$, where $x \in \{0,1\}^{\ell(\lambda)}$, from S, send the message $(\texttt{committed}, \mathsf{sid})$ to R. Upon command $(\texttt{unveil}, \mathsf{sid})$ from S, send $(\texttt{unveiled}, \mathsf{sid}, x)$ to R (with the matching sid). Several commands $(\texttt{commit})$ or $(\texttt{unveil})$ with the same sid are ignored.

Oblivious Transfer Functionality. The oblivious transfer functionality allows for the receiver party to select a bit b and the sender party to send two messages m_0 and m_1 to the receiver in such a way that, the sender never learns the bit b the receiver chose, and the receiver learns only the message m_b, and nothing else about m_{b-1}.

Definition 16 (Oblivious Transfer (OT))**.** Let R and S be two parties. The functionality $\mathcal{F}_{\mathsf{OT}}^{S\to R,\ell}$ behaves as follows: Upon receiving the command $(\textit{transfer}, \mathsf{id}, m_0, m_1)$ from S, with $m_0, m_1 \in \{0,1\}^{\ell(\lambda)}$, send the message $(\textit{received}, \mathsf{id})$ to R; party R replies with $(\textit{choice}, \mathsf{id}, b)$, for $b \in \{0,1\}$. Upon receiving $(\textit{choice}, \mathsf{id}, b)$ from R, send $(\textit{eliver}, \mathsf{id}, m_b)$ to R. We call S the sender, and R the receiver.

Remark 17. Looking ahead, we note that we cannot define the protocol of Section 6 in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model or in the $\mathcal{F}_{\mathsf{MCOM}}$-hybrid model. The former is due to the protocol of Section 6 requiring an OT with the additional property of statistical receiver privacy, which is not the case of all OT protocols that realize the $\mathcal{F}_{\mathsf{OT}}$ functionality. The latter is due to the protocol requiring a commitment scheme with the additional property of statistical hiding, which is not the case of all commitment schemes that realize the $\mathcal{F}_{\mathsf{MCOM}}$ functionality. Moreover, the protocol of Section 6 requires to prove statements about the contents inside of a commitment, and as shown by [CLOS02] this is not possible using a UC commitment functionality.

4 Physical Assumptions

The functionality $\mathcal{F}_{\mathsf{HToken}}$ described in this section models generic *fully* malicious hardware tokens, including PUFs. A fully malicious hardware token is the one that its state is *not* bounded a-priori, its creator can install arbitrary code inside of it, and it can encapsulate an arbitrary number of (possibly fully malicious) tokens inside of itself, called children. As far as we know, this is the

first functionality to integrate tamper-proof hardware tokens with PUFs, allowing us to design protocols that are transparent about the type of hardware token used, as the functionality can be instantiated with any of the former. Moreover, in the particular case of PUFs, our model extends the PUFs-inside-PUF model of [BKOV17] to the more general case of Tokens-inside-Token.[5] We handle encapsulated tokens in the functionality by allowing the *parent* token (i.e., the token that contains other token(s)) to have oracle access to all its children during its evaluation; we believe that token encapsulation models a realistic capability of an adversary and we believe that it is important to include it in our model for the soundness of the security analysis. We also note that $\mathcal{F}_{\mathsf{HToken}}$ is not PPT; this is because the functionality does not impose a restriction on the efficiency of the malicious code.

The functionality $\mathcal{F}_{\mathsf{HToken}}$ allows tokens to be transferred among parties by invoking `handover`; a token can only be queried by the party that currently owns the token by invoking `query`. Malicious tokens can be created by the adversary and it can contain other tokens inside of it. In contrast to [CGS08], the adversary can "unwrap" encapsulated tokens by invoking `openup` and read malicious tokens' state by invoking `readout`.

Functionality $\mathcal{F}_{\mathsf{HToken}}$

$\mathcal{F}_{\mathsf{HToken}}$ is parameterized by an algorithm HTSamp, a PPT Turing machine M_{honest} and a polynomial $p(\lambda)$ that bounds the running time of M_{honest}. $\mathcal{F}_{\mathsf{HToken}}$ runs on input the security parameter 1^λ, with parties $\mathcal{P} = \{P_1, \cdots, P_n\}$, and adversary $\mathcal{A}$. The list $\mathcal{L}$ contains instances of tokens with the attributes id, st, M, children, owner, honest, that can be accessed with the notation *token.attribute*, and where id is a string that uniquely identifies a physical instance of the hardware token, st is the internal state of the token, M is a TM that contains the code to be executed, children is a list of children (tokens) that are contained within this token (can also be empty), owner is the party that currently owns the token (can be *embedded* in case of children), and honest is a boolean value that is true when the token was honestly generated, and false otherwise. For simplicity we omit the polynomial $p(\lambda)$, since wlog any $p(\lambda)$ can be considered. We note that $\mathcal{F}_{\mathsf{HToken}}$ is *not* PPT, and this is due to the fact that there is no runtime bound on M. The functionality $\mathcal{F}_{\mathsf{HToken}}$ receives commands and acts as follows.

- Upon command (`create`) from $P \in \mathcal{P}$, create an empty token tok and do:
  - tok.id $\leftarrow_{\$} \{0,1\}^\lambda$, tok.honest := true, tok.owner := P, tok.children := $\varnothing$, tok.M := M_{honest}, and (tok.st, pubinfo) $\leftarrow$ HTSamp(1^λ).
  - Add tok to $\mathcal{L}$ and return (id, pubinfo) to P.
- Upon command (`createmal`, M, st, children) from $\mathcal{A}$ do: For all tok$_c \in$ children if tok$_c$.owner $= \mathcal{A}$ then,
  - Create an empty token tok, and set tok.id $\leftarrow_{\$} \{0,1\}^\lambda$, tok.honest := false, tok.owner := $\mathcal{A}$, tok.M := M, tok.children := children, and tok.st := st.
  - Add tok to $\mathcal{L}$, and for all tok$_c \in$ children set tok$_c$.owner := *embedded*.
  - Return tok.id to $\mathcal{A}$.
- Upon command (`handover`, id, P_j) from $P_i \in \mathcal{P} \cup \{\mathcal{A}\}$, where $P_j \in \mathcal{P} \cup \{\mathcal{A}\}$: For all tok $\in \mathcal{L}$ s.t. tok.owner $= P_i$ and tok.id = id do.
  - Set tok.owner := P_j.
  - Send (`handover`, id, P_i) to P_j.
- Upon command (`query`, id, q) from $P \in \mathcal{P} \cup \{\mathcal{A}\}$: Define the recursive algorithm HTEval as follows.
  - $a \leftarrow$ HTEval$(1^\lambda$, id, $q)$: It takes as input the security parameter, the id of the hardware

[5] In the model of [BKOV17] the malicious PUFs-inside-PUF are stateless, while $\mathcal{F}_{\mathsf{HToken}}$ allows the malicious PUFs to be stateful.

token, and a challenge q. It first runs $(a, \mathsf{st}') \leftarrow \mathsf{M}(\mathsf{st}, q)$, where M is interpreted as the code for an oracle Turing machine with $|\mathsf{children}|$ oracles. When M makes an oracle query q' to its i-th oracle, run $b \leftarrow \mathsf{HTEval}(1^\lambda, \mathsf{id}_i, q')$ recursively and answer the query with b. M updates its state st to the new state st' after its execution. Return a.
  - For all $\mathsf{tok} \in \mathcal{L}$ s.t. $\mathsf{tok.id} = \mathsf{id}$, and $\mathsf{tok.owner} = P$ do: Run $a \leftarrow \mathsf{HTEval}(1^\lambda, \mathsf{tok.id}, q)$ and send a to party P.
- Upon command $(\texttt{readout}, \mathsf{id})$ from $\mathcal{A}$: For all $\mathsf{tok} \in \mathcal{L}$ s.t. $\mathsf{tok.id} = \mathsf{id}$, $\mathsf{tok.owner} = \mathcal{A}$, and $\mathsf{tok.honest} = \mathsf{false}$ do.
  - Return $\mathsf{tok.st}$ to $\mathcal{A}$
- Upon command $(\texttt{openup}, \mathsf{id})$ from $\mathcal{A}$: For all $\mathsf{tok} \in \mathcal{L}$ s.t. $\mathsf{tok.id} = \mathsf{id}$, $\mathsf{tok.owner} = \mathcal{A}$, and $\mathsf{tok.honest} = \mathsf{false}$ do.
  - Remove tok from $\mathcal{L}$, and for each $\mathsf{tok}_c \in \mathsf{tok.children}$:
    - Set $\mathsf{tok}_c.\mathsf{owner} := P$, for some $P \in \mathcal{P}$.
  - Return $\texttt{ok}$ to $\mathcal{A}$.
- In all other cases, enter the waiting state without sending a message.

The long-term output tape a records all the information from the tokens in $\mathcal{L}$ such that $\mathsf{tok.owner} = \mathcal{A}$ (or tokens owned by some other token that is owned by $\mathcal{A}$, for any number of layers).

4.1 Physically Uncloneable Functions (PUFs)

In a nutshell, a PUF is a noisy source of randomness. It is a hardware device that, upon physical stimuli, called *challenges*, produces physical outputs (that are measured), called *responses*. The response measured for each challenge of the PUF is unpredictable, in the sense that it is hard to *predict* the response of the PUF on a given challenge without first measuring the response of the PUF on the same (or similar) challenge. When a PUF receives the same physical stimulus more than once, the responses produced may not be exactly equal (due to the added noise), but the Hamming distance of the responses are bounded by a parameter of the PUF.

A family of PUFs is a pair of algorithms $(\mathsf{PUFSamp}, \mathsf{PUFEval})$, not necessarily PPT. $\mathsf{PUFSamp}$ models the manufacturing process of the PUF: On input the security parameter, it draws an index σ, that represents an instance of a PUF that satisfies the security definitions for the security parameter (that we define later). $\mathsf{PUFEval}$ models a physical stimulus applied to the PUF: Upon a challenge input x, it invokes the PUF with x and measures the response y, that is returned as the output. The length of a response y returned by algorithm $\mathsf{PUFEval}$ is a bitstring of size rg. A formal definition follows.

Definition 18 (Physically Uncloneable Functions)**.** Let rg denote the size (in bits) of the range of the PUF responses of a PUF family. The pair $\mathsf{PUF} = (\mathsf{PUFSamp}, \mathsf{PUFEval})$ is a PUF family if it satisfies the following properties.

- *Sampling.* Let $\mathcal{I}_\lambda$ be an index set. On input the security parameter λ, the stateless and unbounded sampling algorithm $\mathsf{PUFSamp}$ outputs an index $\sigma \in \mathcal{I}_\lambda$. Each $\sigma \in \mathcal{I}_\lambda$ corresponds to a family of distributions $\mathcal{D}_\sigma$. For each challenge $x \in \{0,1\}^\lambda$, $\mathcal{D}_\sigma$ contains a distribution $\mathcal{D}_\sigma(x)$ on $\{0,1\}^{\mathsf{rg}(\lambda)}$. It is not required that $\mathsf{PUFSamp}$ is a PPT algorithm.
- *Evaluation.* On input $(1^\lambda, \sigma, x)$, where $x \in \{0,1\}^\lambda$, the evaluation algorithm $\mathsf{PUFEval}$ outputs a response $y \in \{0,1\}^{\mathsf{rg}(\lambda)}$ according to the distribution $\mathcal{D}_\sigma(x)$. It is not required that $\mathsf{PUFEval}$ is a PPT algorithm.

Additionally, we require the PUF family to satisfy a reproducibility notion that we describe next. Reproducibility informally says that, the responses produced by the PUF when queried on the same random challenge are always close.

Definition 19 (PUF Reproducibility)**.** A PUF family $\mathsf{PUF} = (\mathsf{PUFSamp}, \mathsf{PUFEval})$, for security parameter λ, is δ-reproducible if for $\sigma \leftarrow \mathsf{PUFSamp}(1^\lambda)$, $x \leftarrow\!\!\$\ \{0,1\}^\lambda$, and $y \leftarrow \mathsf{PUFEval}(\sigma, x)$, $y' \leftarrow \mathsf{PUFEval}(\sigma, x)$, we have that,

$$\Pr\left[\mathsf{hd}(y, y') \leq \delta\right] \geq 1 - \mathsf{negl}(\lambda),$$

for a negligible function $\mathsf{negl}(\lambda)$.

Many PUF definitions in the literature [OSVW13, BKOV17, BFSK11, DFK+14] have had problems with the super-polynomial nature of PUFs. In particular, the possibility of PUFs solving hard computational problems, such as discrete logarithms or factoring, was not excluded, or excluded in an awkward way. We take our inspiration from the idea that a PUF can be thought as a function selected at random from a very large set, and therefore cannot be succinctly described; however, it can be efficiently simulated using lazy sampling. Conceptually, we will only consider PUFs that can be efficiently simulated by a stateful machine.

Definition 20. A polynomial-time (stateful) interactive Turing machine $(\mathsf{MSamp}, \mathsf{MEval})$ is a *lazy sampler* for $(\mathsf{PUFSamp}, \mathsf{PUFEval})$ such that for all sequences $(x_1, \ldots, x_n)$ of inputs, the random variables $(Y_1, \ldots, Y_n)$ and $(Y'_1, \ldots, Y'_n)$, defined by the following experiments, are identically distributed.

$$\mathsf{st} \leftarrow \mathsf{MSamp}(1^\lambda); Y_1 \leftarrow \mathsf{MEval}(x_1), \ldots, Y_n \leftarrow \mathsf{MEval}(x_n);$$
$$\sigma \leftarrow \mathsf{PUFSamp}(1^\lambda); Y'_1 \leftarrow \mathsf{PUFEval}(\sigma, x_1), \ldots, Y'_n \leftarrow \mathsf{PUFEval}(\sigma, x_n);$$

where st denotes the initial state of the TM M.

Security of PUFs. The security of PUFs have been mainly defined by the properties of unpredictability and uncloneability [Mae13, OSVW13, BKOV17, BFSK11, AMSY16]. In Section 4.1.1 we introduce a novel unpredictability notion for PUFs, and we later discuss why the standard unpredictability notion is not suited for our setting.

4.1.1 Fully-adaptive PUF Unpredictability.

In contrast to the standard definition of unpredictability [BFSK11], in this work we require a stronger notion of adaptive unpredictability. Loosely speaking, unpredictability should capture the fact that it is hard to learn the response of the PUF on a given challenge without first querying the PUF on a similar challenge. Note that this implies uncloneability: If one could clone the PUF, one could use the cloned PUF to predict the answers of the original PUF. We express the similarity of inputs/outputs of the PUF in terms of the Hamming distance hd, however, our results can be easily adapted to other metrics.

Definition 21 (Adaptive PUF Unpredictability)**.** A PUF family $\mathsf{PUF} = (\mathsf{PUFSamp}, \mathsf{PUFEval})$, for security parameter λ, is (γ, δ)-unpredictable if for all adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$, such that,

$$Pr\left[(\mathsf{hd}(y, y') \leq \delta) \wedge (\mathsf{hd}(q, x) \geq \gamma,\ \forall q \in \mathcal{Q}) : \begin{array}{c} \sigma \leftarrow \mathsf{PUFSamp}(1^\lambda); \\ x \leftarrow\!\!\$\ \{0,1\}^\lambda; \\ y \leftarrow \mathcal{A}^{\mathsf{PUFEval}(1^\lambda, \sigma, \cdot)}(x); \\ y' \leftarrow \mathsf{PUFEval}(1^\lambda, \sigma, x); \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where $\mathcal{Q}$ is the list of all queries made by $\mathcal{A}$.

The adaptive PUF unpredictability says that the only way to learn the output of $\mathsf{PUFEval}(1^\lambda, \sigma, x)$ is to query the PUF on x (or something close enough to x). Our definition captures this by allowing adversary $\mathcal{A}$ to know the challenge x *before* having oracle access to $\mathsf{PUFEval}$.

The unsuitability of the standard PUF unpredictability of [BFSK11]. We first recall the standard unpredictability definition of [BFSK11]. As the definition itself is based on the notion of average min-entropy, for convenience, we present that first.

Definition 22 (Average Min-entropy [BFSK11])**.** The average min-entropy of the measurement $\mathsf{PUFEval}(q)$ conditioned on the measurements of challenges $\mathcal{Q} = \{q_1, \cdots, q_{\mathsf{poly}(\lambda)}\}$ for the PUF family $\mathsf{PUF} = (\mathsf{PUFSamp}, \mathsf{PUFEval})$ is defined by

$$\begin{aligned}
&\tilde{\mathsf{H}}_\infty(\mathsf{PUFEval}(q)|\mathsf{PUFEval}(\mathcal{Q})) = \\
&= -\log\left(\mathbb{E}_{a_k \leftarrow \mathsf{PUFEval}(q_k)}\left[\max_a \Pr\left[\mathsf{PUFEval}(q) = a \;\middle|\; a_1 = \mathsf{PUFEval}(q_1), \cdots, a_{\mathsf{poly}(\lambda)} = \mathsf{PUFEval}(q_{\mathsf{poly}(\lambda)})\right]\right]\right) \\
&= -\log\left(\mathbb{E}_{a_k \leftarrow \mathsf{PUFEval}(q_k)}\left[2^{\mathsf{H}_\infty(\mathsf{PUFEval}(q)|a_1 = \mathsf{PUFEval}(q_1), \cdots, a_{\mathsf{poly}(\lambda)} = \mathsf{PUFEval}(q_{\mathsf{poly}(\lambda)}))}\right]\right)
\end{aligned}$$

where the probability is taken over the choice of σ from $\mathcal{I}_\lambda$ and the choice of possible PUF responses on challenge q. The term $\mathsf{PUFEval}(\mathcal{Q})$ denotes a sequence of random variables $\mathsf{PUFEval}(q_1)$, $\cdots, \mathsf{PUFEval}(q_{\mathsf{poly}(\lambda)})$, each corresponding to an evaluation of the PUF on challenge q_k, for $1 \geq k \geq \mathsf{poly}(\lambda)$.

Definition 23 (PUF Unpredictability [BFSK11])**.** A (rg, δ)-PUF family $\mathsf{PUF} = (\mathsf{PUFSamp}, \mathsf{PUFEval})$ for security parameter λ is $(\gamma(\lambda), \mathsf{m}(\lambda))$-unpredictable if for any $q \in \{0,1\}^\lambda$ and challenge list $\mathcal{Q} = \{q_1, \ldots, q_{\mathsf{poly}(\lambda)}\}$, one has that, if for all $1 \geq k \geq \mathsf{poly}(\lambda)$ the Hamming distance satisfies $\mathsf{hd}(q, q_k) \geq \gamma(\lambda)$, then the average min-entropy satisfies $\tilde{\mathsf{H}}_\infty(\mathsf{PUFEval}(q)|\mathsf{PUFEval}(\mathcal{Q})) \geq \mathsf{m}(\lambda)$, where $\mathsf{PUFEval}(\mathcal{Q})$ denotes the sequence of random variables $\mathsf{PUFEval}(q_1), \cdots, \mathsf{PUFEval}(q_{\mathsf{poly}(\lambda)})$, each corresponding to an evaluation of the PUF on challenge q_k. Such a PUF family is called a $(\mathsf{rg}, \delta, \gamma, \mathsf{m})$-PUF family.

We now argue why Definition 23 is not suited for our setting. We present a PUF family that satisfies Definition 23 and yet allows for an adversary to predict the response of the PUF on a challenge never queried to the PUF (and far apart from the other queried challenges). We prove the following theorem next.

Theorem 24. *There exists a PUF family* $\mathsf{PUF} = (\mathsf{PUFSamp}, \mathsf{PUFEval})$ *that satisfies Definition 23 (with* $\mathsf{m} > 0$*), such that there exists a PPT adversary* $\mathcal{A}$ *that can predict with probability* 1 *the output of the PUF on an input far from every other input queried to the PUF prior (thereby contradicting Definition 21).*

PROOF. Let $\mathsf{PUF} = (\mathsf{PUFSamp}, \mathsf{PUFEval})$ be a PUF family for challenges of size $(n+1)$-bits and responses of size n-bits, We construct the family PUF as follows:

- $\mathsf{PUFSamp}(1^\lambda)$: Samples $x^* \leftarrow\!\!\$\ \{0,1\}^n$, and $f \leftarrow\!\!\$\ (\{0,1\}^n \to \{0,1\}^n)$. Return $\sigma := (x^*, f)$.
- $\mathsf{PUFEval}(1^\lambda, \sigma, x)$:
  - Upon query $\mathsf{PUFEval}(1^\lambda, \sigma, 0^{n+1})$ output x^*.
  - Upon query $\mathsf{PUFEval}(1^\lambda, \sigma, 0\|m)$ with $m \neq 0^n$ output $f(m)$.
  - Upon query $\mathsf{PUFEval}(1^\lambda, \sigma, 1\|m)$ output $f(m \oplus x^*)$.

We first show how an adversary can predict with probability 1 the output of a PUF from the family described above on a fresh input. Given some arbitrary fresh challenge input $b\|m$, the adversary can find the corresponding response $\mathsf{PUFEval}(1^\lambda, \sigma, b\|m)$, *without ever querying* the PUF on $b\|m$, by doing the following: Compute $x^* := \mathsf{PUFEval}(1^\lambda, \sigma, 0^{n+1})$ and compute $y := \mathsf{PUFEval}(1^\lambda, \sigma, \bar{b}\|m \oplus x^*)$. Note that both queries are far apart from $b\|m$, yet the adversary learns $y = \mathsf{PUFEval}(1^\lambda, \sigma, b\|m) = \mathsf{PUFEval}(1^\lambda, \sigma, \bar{b}\|m \oplus x^*)$.

Now we show that the PUF family described above satisfies Definition 23.[6] Fix any polynomial-size challenge list $\mathcal{Q} = \{q_1, \ldots, q_{\kappa-1}\}$ and any challenge query q_κ such that, for any $k \in [\kappa - 1]$: $\mathsf{hd}(q_\kappa, q_k) \geq 1$, which is clearly minimal. Since f is a random function, it holds that $\mathsf{PUFEval}(1^\lambda, \sigma, q)$ has maximal average min-entropy, unless the PUF is queried on two inputs (q_i, q_j) that form a collision for f. Note that this happens only if $q_i \oplus q_j = 1\|x^*$. Thus all we need to show is that, for any fixed set of queries $\{q_1, \ldots, q_\kappa\}$ the probability that $q_i \oplus q_j = 1\|x^*$ is negligible, over the random choice of x^*. This holds because

$$\Pr_{x^* \leftarrow \{0,1\}^n}[\exists\ (i,j) : q_i \oplus q_j = 1\|x^*] = 1 - \left(1 - 2^{-n}\right)^{\kappa^2} \leq 1 - \frac{1}{1 + \kappa^2 2^{-n}} = \frac{\kappa^2 2^{-n}}{1 + \kappa^2 2^{-n}}$$

by applying the Bernoulli inequality. The above expression approaches 0 exponentially fast, as n grows. This concludes our proof. □

Contrasting our unpredictability definition with the one of [BFSK11]. The motivation behind our newly proposed adaptive unpredictability notion (Definition 21), is that the standard PUF unpredictability notion of [BFSK11] *implicitly assumes* that PUFs are only dependent on random physical factors (likely introduced during manufacturing), and in particular it does not capture families of PUFs that could have some programmability built in, allowing to predict the output of a PUF on an input by querying a completely different input. What our new PUF unpredicatability notion explicitly captures is that a "good" PUF *must* solely depend on random physical factors, and in particular cannot have any form of programmability. On a more philosophical level, we believe that our new notion is what was meant to be modelled as a property for PUFs from the start. Since PUFs are inherently randomized devices that are specifically built to be unpredictable and uncontrollable, a PUF family such as the one described above should not be considered to be a "good" PUF family; however the previous notion fails to capture this fact.[7]

Overall, our new definition of unpredictability does not hinder in any way the progress and development of new real-world PUFs, but merely addresses a technical oversight by the previous

[6] This unpredictability definition is considered in many previous works, such as [BFSK11, OSVW13, BKOV17, Mae13].

[7] The authors of [BFSK11] discuss in Appendix C the different notions of security for PUFs and their relationships, and in particular mention that their definition of unpredictability assumes that the creation process of the PUF is not controllable.

unpredicatability notion. Therefore, we conjecture that most real-world PUFs that satisfy the unpredictability notion of [BFSK11] will most likely also satisfy our unpredicatability notion, since real PUFs are inherently randomized physical devices built to be unpredictable and uncontrollable.

5 Impossibility of Everlasting OT with Malicious Hardware

In this section we prove the impossibility of realizing everlasting secure oblivious transfer (OT) in the hardware token model, *even in the presence of a trusted setup*. The result carries over immediately to any secure computation protocol due to the completeness of OT [Kil88]. We consider honest tokens to be stateful but non-erasable (Definition 25) and the tokens produced by the adversary can be malicious but not encapsulate other tokens (note that this restriction on malicious tokens only makes our result stronger, as the impossibility holds even against an adversary that is more limited). The adversary $\mathcal{A}$ is PPT during the execution of the protocol, but $\mathcal{A}$ becomes unbounded after the execution is over (i.e. everlasting security). This extends the seminal result of Goyal et al. [GIMS10] that shows the impossibility of having *statistically* (as opposed to everlasting) UC secure oblivious transfer from stateless (as opposed to non-erasable) tokens. We stress however, that our negative results does not contradict the work of Döttling et al. [DKMQ11,DKMN15], since they assume honest tokens to be non-resettable or bounded-resettable (i.e., tokens cannot be reset to a previous state, or only reset up to an a-priori bound), whereas for our result to hold the token must be non-erasable.

In the following, we show the main theorem of the section. The result holds under the assumption that the token scheduling is fixed a-priori, which captures most of the known protocols for secure computation [DKMQ11, DKMN15, HPV16]. The scheduling of the tokens determines the exchange of the tokens among parties. We stress that we do not impose any restriction on which party will hold each hardware token in the end of the execution. For a formal definition of OT we refer the reader to Section 2.2. We first define "non-erasability" for hardware tokens next.

Definition 25 (Non-erasable hardware token)**.** A (stateful) hardware token is said to be non-erasable if any state ever recorded by the token can be efficiently retrieved.

Note in particular that stateless tokens are trivially non-erasable, as the former cannot keep any state.

Theorem 26. *Let Π be a hardware token-based everlasting OT protocol between Alice (i.e. sender) and Bob (i.e. receiver) where the honest tokens are non-erasable and the scheduling of the tokens is fixed. Then, at least one of the following holds:*

- *There exists an everlasting adversary $\mathcal{S}'$ that uses malicious and stateful hardware tokens such that $\mathbf{Adv}^{\Pi}_{\mathcal{S}'} \geq \epsilon(\lambda)$, or*
- *there exists an everlasting adversary $\mathcal{R}'$ that uses malicious and stateful hardware tokens such that $\mathbf{Adv}^{\Pi}_{\mathcal{R}'} \geq \epsilon(\lambda)$,*

for some non-negligible function $\epsilon(\lambda)$.

Proof. The proof consists of the following sequence of modified simulations. Let the game $\mathfrak{G}_0$ define an everlastingly-secure OT protocol for $\mathcal{S}$ and $\mathcal{R}$. Then, by assumption, we have that for all everlasting adversaries $\mathcal{S}'$ and $\mathcal{R}'$ it holds that

$$\mathbf{Adv}^{\mathfrak{G}_0}_{\mathcal{S}'} \leq \mathsf{negl}(\lambda) \text{ and } \mathbf{Adv}^{\mathfrak{G}_0}_{\mathcal{R}'} \leq \mathsf{negl}(\lambda).$$

We define a *quasi-semi-honest* adversary to be an adversary that behaves semi-honestly but keeps a log of all queries ever made to the hardware token (i.e. the non-erasable token assumption). Let the game $\mathfrak{G}_1$ define an everlastingly-secure OT protocol where $\mathcal{S}'$ and $\mathcal{R}'$ are quasi-semi-honest. Since we are strictly reducing the capabilities of the adversaries and the tokens are non-erasable, we can state the following lemma.

Lemma 27. *For all quasi-semi-honest $\mathcal{S}'$ and $\mathcal{R}'$ it holds that*

$$\mathbf{Adv}^{\mathfrak{G}_1}_{\mathcal{S}'} \leq \mathsf{negl}(\lambda) \text{ and } \mathbf{Adv}^{\mathfrak{G}_1}_{\mathcal{R}'} \leq \mathsf{negl}(\lambda).$$

Let $\mathfrak{G}_2$ be the same as $\mathfrak{G}_1$ except that whenever $\mathcal{S}'$ (resp. $\mathcal{R}'$) queries a token from $\mathcal{R}$ (resp. $\mathcal{S}$) that *will return* to $\mathcal{R}$ (resp. $\mathcal{S}$), instead of making that query to the token, $\mathcal{S}'$ queries directly $\mathcal{R}$ who answers it as the token would have. Since the distribution of the answers for the queries does not change, we can state the following lemma.

Lemma 28. *For all quasi-semi-honest $\mathcal{S}'$ and $\mathcal{R}'$ it holds that*

$$\mathbf{Adv}^{\mathfrak{G}_2}_{\mathcal{S}'} \leq \mathsf{negl}(\lambda) \text{ and } \mathbf{Adv}^{\mathfrak{G}_2}_{\mathcal{R}'} \leq \mathsf{negl}(\lambda).$$

Let game $\mathfrak{G}_3$ be exactly the same as $\mathfrak{G}_2$ except that whenever $\mathcal{S}'$ (resp. $\mathcal{R}'$) sends a token to $\mathcal{R}$ (resp. $\mathcal{S}$) that *will not return* to $\mathcal{S}'$ (resp. $\mathcal{R}'$), then $\mathcal{S}'$ sends a description of the token instead. Since we consider everlasting adversaries we assume that after the execution of the protocol all tokens can be read out. Therefore both parties will have the description of all the tokens, even the ones that are not sent to the other party. Note that at this point there are no hardware tokens involved, and only description of tokens. Therefore a quasi-semi-honest adversary is identical to a semi-honest everlasting one.[8]

Lemma 29. *For all semi-honest everlasting $\mathcal{S}'$ and $\mathcal{R}'$ it holds that*

$$\mathbf{Adv}^{\mathfrak{G}_3}_{\mathcal{S}'} \leq \mathsf{negl}(\lambda) \text{ and } \mathbf{Adv}^{\mathfrak{G}_3}_{\mathcal{R}'} \leq \mathsf{negl}(\lambda).$$

We point out that a semi-honest unbounded adversary $\mathcal{S}'$ (resp. $\mathcal{R}'$) is also a semi-honest everlasting adversary, since during the execution of the protocol it performs only the honest (PPT) actions. We are now in the position of stating the final lemma.

Lemma 30. *For all semi-honest unbounded $\mathcal{S}'$ and $\mathcal{R}'$ it holds that*

$$\mathbf{Adv}^{\mathfrak{G}_3}_{\mathcal{S}'} \leq \mathsf{negl}(\lambda) \text{ and } \mathbf{Adv}^{\mathfrak{G}_3}_{\mathcal{R}'} \leq \mathsf{negl}(\lambda).$$

It was shown [Bea96] that it is not possible to build a secure OT protocol against semi-honest unbounded adversaries (even in the presence of a trusted setup), what gives us a contradiction and concludes our proof. □

[8] An everlasting semi-honest adversary follows the protocol honestly, but can behave arbitrarily after the protocol runs is over and it becomes unbounded.

6 Everlasting Commitment from Fully Malicious PUFs

In this section, we build an everlastingly secure UC commitment scheme from fully malicious PUFs. Let $\mathcal{C} = (\mathsf{Com}, \mathsf{Open})$ be a statistically hiding UC-secure commitment scheme, let $(\mathsf{Sender}_{\mathsf{OT}}, \mathsf{Receiver}_{\mathsf{OT}})$ be a 1-out-of-2 statistically receiver-private UC-secure OT, let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a one-way permutation, and let $H : \{0,1\}^{d(\lambda)} \times \{0,1\}^{\ell(\lambda)} \to \{0,1\}^c$ be a strong randomness extractor, where d and ℓ are two polynomials such that H allows for $(\ell(\lambda) - c)$-many bits of entropy loss, for $c := |m \| \mathsf{decom}|$. Let

$$R_1 := \left\{ \begin{array}{c} \left(y, \{\mathsf{com}_i\}_{i\in[\ell(\lambda)]}\right), \\ \left(\{m_i, \mathsf{decom}_i\}_{i\in[\ell(\lambda)]}, x\right) \end{array} \middle| \begin{array}{c} y = f(x) \vee \\ \{m_i = \mathsf{Open}(\mathsf{decom}_i, \mathsf{com}_i)\}_{i\in[\ell(\lambda)]} \end{array} \right\}$$

and let

$$R_2 := \left\{ \begin{pmatrix} y, \mathsf{seed}, m, \mathsf{com}, \omega, \\ \{\mathsf{com}_i, q_i^0, q_i^1\}_{i\in[\ell(\lambda)]} \end{pmatrix} \middle| \begin{pmatrix} y = f(x) \vee \\ m \| \mathsf{decom} = H(\mathsf{seed}, k) \oplus \omega \wedge \\ m = \mathsf{Open}(\mathsf{com}, \mathsf{decom})) \wedge \\ \left\{ \begin{array}{c} s_i = \mathsf{Open}(\mathsf{decom}_i, \mathsf{com}_i) \\ \wedge \mathsf{hd}(s_i, q_i^{k_i}) \leq \delta \end{array} \right\}_{i\in[\ell(\lambda)]} \end{pmatrix} \right\}.$$

We denote by $(\mathcal{P}_1, \mathcal{V}_1)$ and $(\mathcal{P}_2, \mathcal{V}_2)$ the statistically witness-indistinguishable arguments of knowledge (SWIAoK) for the relations R_1 and R_2, respectively. Our commitment scheme is described next.

Everlasting Commitment scheme from PUF

SETUP: Let $\mathcal{G}$ be the distribution for a random y in the range of the one-way permutation f, let seed be a random seed for the strong randomness extractor H, and let $\mathsf{crs}_{\mathsf{com}}$ be the CRS for the non-interactive commitment and $\mathsf{crs}_{\mathsf{OT}}$ for the OT protocol. The ideal functionality $\mathcal{F}_{\mathsf{CRS}}^{\mathcal{G}}$ samples a random crs from the distribution of valid values, where $\mathsf{crs} := (y, \mathsf{seed}, \mathsf{crs}_{\mathsf{com}}, \mathsf{crs}_{\mathsf{OT}})$ and provides Alice and Bob with crs. We denote by $x \in \{0,1\}^\lambda$ the pre-image such that $f(x) = y$, used by $\mathcal{F}_{\mathsf{CRS}}^{\mathcal{G}}$ to sample y.

COMMITMENT: On input (*commit*, id, m), for a fresh id, Alice engages with Bob in the following interactive protocol.

1. Bob samples a PUF token by querying $\mathcal{F}_{\mathsf{HToken}}^{\mathsf{PUFEval},\mathsf{PUFSamp}}$ on message (`create`), and receives back $(\mathsf{id}_{\mathsf{PUF}}, \mathsf{pubinfo})$; it then samples $\ell(\lambda)$-many random tuples of the form $(p_i^0, p_i^1) \in \{0,1\}^{2\lambda}$ and queries $\mathcal{F}_{\mathsf{HToken}}^{\mathsf{PUFEval},\mathsf{PUFSamp}}$ on all pairs to obtain (q_i^0, q_i^1). Bob finally transfers the token to Alice by querying $(\mathtt{handover}, \mathsf{id}_{\mathsf{PUF}}, \mathit{Alice})$.
2. Alice samples a random string $k \leftarrow \{0,1\}^{\ell(\lambda)}$, and computes $(\mathsf{com}, \mathsf{decom}) \leftarrow \mathsf{Com}(m)$. Then for all $i \in [\ell(\lambda)]$ she engages with Bob in an oblivious transfer protocol on input $\alpha_i \leftarrow \mathsf{Receiver}_{\mathsf{OT}}(k_i)$. All messages of the oblivious transfer are tagged with id.
3. Bob responds to each i-th instance of the oblivious transfer with $\mathsf{Sender}_{\mathsf{OT}}(p_i^0, p_i^1)$.
4. For all $i \in [\ell(\lambda)]$ Alice queries $\mathcal{F}_{\mathsf{HToken}}^{\mathsf{PUFEval},\mathsf{PUFSamp}}$ on α_i and parses the response as β_i. If the token does not return a valid output, Alice sets $\beta_i = 0^{\ell(\lambda)}$. Alice then commits to $(\mathsf{com}_i, \mathsf{decom}_i) \leftarrow \mathsf{Com}(\beta_i)$ and sends the tuple $(\mathsf{id}, \mathsf{com}, \omega := H(\mathsf{seed}, k) \oplus m \| \mathsf{decom}, \{\mathsf{com}_i\}_{i\in[\ell(\lambda)]})$ to Bob. Finally, Alice interacts with Bob with the algorithm $\mathcal{P}_1((y, \{\mathsf{com}_i\}_{i\in[\ell(\lambda)]}), (\{\beta_i, \mathsf{decom}_i\}_{i\in[\ell(\lambda)]}, 0))$. All messages of the SWIAoK are tagged with id.
5. Bob executes $\mathcal{V}_1(y, \{\mathsf{com}_i\}_{i\in[\ell(\lambda)]})$ and aborts all interactions with Alice if the algorithm does not return 1, including other instances of this commitment protocol. Otherwise Bob sends

$(\mathsf{id}, \{(q_i^0, q_i^1)\}_{i\in[\ell(\lambda)]})$ to Alice and outputs $(\textit{commited}, \mathsf{id})$.

OPENING: On input $(\textit{unveil}, \mathsf{id})$, Alice parses $(\mathsf{com}, \mathsf{decom}, \omega, k, \{\mathsf{com}_i\}_{i\in[\ell(\lambda)]}, \{\mathsf{decom}_i\}_{i\in[\ell(\lambda)]}, \{q_i^0, q_i^1\}_{i\in[\ell(\lambda)]})$ as the information generated in the commitment phase with the same id, if any, and m as the corresponding message. Then it interacts with Bob in the following manner.

1. In the opening phase Alice sends m to Bob and executes $\mathcal{P}_2((y, \mathsf{seed}, m, \mathsf{com}, \omega, \{\mathsf{com}_i\}_{i\in[\ell(\lambda)]}, \{q_i^0, q_i^1\}_{i\in[\ell(\lambda)]}), (k, \{\mathsf{decom}_i\}_{i\in[\ell(\lambda)]}, \mathsf{decom}, 0))$ in interaction with Bob.
2. Bob receives m and runs $\mathcal{V}_2(y, \mathsf{seed}, m, \mathsf{com}, \omega, \{\mathsf{com}_i\}_{i\in[\ell(\lambda)]}, \{q_i^0, q_i^1\}_{i\in[\ell(\lambda)]})$ in interaction with Alice. If the protocol returns 1, then Bob returns $(\textit{unveiled}, \mathsf{id}, m)$.

We note that many instances of the previously described protocol (with a different id) may run concurrently.

Theorem 31. *Let*

- $\mathcal{C} = (\mathsf{Com}, \mathsf{Open})$ *be a statistically hiding and computationally binding commitment scheme,*
- $(\mathsf{Sender}_{\mathsf{OT}}, \mathsf{Receiver}_{\mathsf{OT}})$ *be a UC-secure* 1*-out-of-*2 *statistically receiver-private oblivious transfer,*
- $H : \{0,1\}^{d(\lambda)} \times \{0,1\}^{\ell(\lambda)} \to \{0,1\}^c$ *be a strong randomness extractor, where* d *and* ℓ *are two polynomials such that* H *allows for* $(\ell(\lambda) - c)$*-many bits of entropy loss,*
- *and let* $(\mathcal{P}_1, \mathcal{V}_1)$ *and* $(\mathcal{P}_2, \mathcal{V}_2)$ *be SWIAoK systems for the relations* R_1 *and* R_2*, respectively.*

Then the protocol above everlastingly UC-realizes the functionality $\mathcal{F}_{MCOM}$ *in the* $\mathcal{F}_{HToken}^{\mathsf{PUFEval},\mathsf{PUFSamp}}$*-hybrid model.*

PROOF. We consider the cases of the two corrupted parties separately. The proof consists of the description of a series of hybrids and we argue about the indistinguishability of neighbouring experiments. Then we describe a simulator that reproduces the real-world protocol to the corrupted party while executing the protocol in interaction with the ideal functionality.

Corrupted Bob (recipient). Consider the following sequence of hybrids, with $\mathcal{H}_0$ being the protocol as defined above in interaction with $\mathcal{A}$ and $\mathcal{Z}$:

$\mathcal{H}_1$: Defined exactly as in H_0 except that, for all executions of commitment and opening routines, the SWIAoK for R_1 and R_2 are computed using the knowledge of x, the pre-image of y. This is possible as the $\mathcal{F}_{\mathsf{CRS}}$ functionality is simulated by the simulator that samples an $f(x) = y$ such that it knows x. In order to avoid trivial distinguishing attack we additionally require Alice to explicitly check that $\forall i \in \{[\ell(\lambda)]\} : \mathsf{hd}(\beta_i, q_i^{k_i}) \leq \delta$ and abort (prior to computing the SWIAoK) if the condition is not satisfied. The two protocols are statistically indistinguishable due to the statistical witness indistinguishability of the SWIAoK scheme. In particular, for all unbounded distinguishers $\mathcal{D}$ querying the functionality polynomially many times, it holds that

$$\{EXC'_{H_0,\mathcal{A},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N}, z\in\{0,1\}^{poly(\lambda)}} \approx \{EXC'_{H_1,\mathcal{A},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N}, z\in\{0,1\}^{poly(\lambda)}}.$$

$\mathcal{H}_2, \ldots, \mathcal{H}_{\ell(\lambda)+1}$: Each $\mathcal{H}_{1+i}$ for $i \in [\ell(\lambda)]$ is defined exactly as $\mathcal{H}_1$ except that in all of the sessions Alice uses the simulator of the statistical receiver private OT protocol (that implements the $\mathcal{F}_{\mathsf{OT}}$ functionality) to run the first i instances of the oblivious transfers. Note that the simulator (using the knowledge of the CRS trapdoor) returns both of the inputs of the sender, in this case (p_i^0, p_i^1). By statistical receiver privacy of the oblivious transfer, it holds that the simulated execution is

statistically close to a honest run and therefore we have that for all unbounded distinguishers $\mathcal{D}$ that queries $\mathcal{F}_{\mathsf{HToken}}^{\mathsf{PUFEval},\mathsf{PUFSamp}}$ polynomially-many times:

$$\{EXC'_{\mathcal{H}_1,\mathcal{A},\mathcal{D}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}} \approx \{EXC'_{\mathcal{H}_{\ell(\lambda)+1},\mathcal{A},\mathcal{D}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}}.$$

$\mathcal{H}_{\ell(\lambda)+2},\ldots,\mathcal{H}_{2\cdot\ell(\lambda)+1}$: Each $\mathcal{H}_{\ell(\lambda)+1+i}$ for $i\in[\ell(\lambda)]$ is defined exactly as $\mathcal{H}_{\ell(\lambda)+1}$ with the difference that, in all of the sessions, the first i-many commitments com_i are computed as $\mathsf{Com}(r_i)$, for some random r_i in the appropriate domain. Note that the corresponding decommitments are no longer used in the computation of the SWIAoK. Therefore the statistically hiding property of the commitment scheme guarantees that the neighbouring simulations are statistically close for all unbounded $\mathcal{D}$. That is

$$\{EXC'_{\mathcal{H}_{\ell(\lambda)+1},\mathcal{A},\mathcal{D}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}} \approx \{EXC'_{\mathcal{H}_{2\ell(\lambda)+1},\mathcal{A},\mathcal{D}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}}.$$

$\mathcal{H}_{2\cdot\ell(\lambda)+2}$: Let n be a bound on the total number of sessions. The hybrid $\mathcal{H}_{2\cdot\ell(\lambda)+2}$ is defined as the previous except that Alice chooses some random values $(k_1,\ldots,k_n)\in\{0,1\}^{\ell(\lambda)}$ at the beginning of the execution. In the i-th session Alice uses the value of k_i instead of a fresh k in the interaction with the funcitonality $\mathcal{F}_{\mathsf{HToken}}^{\mathsf{PUFEval},\mathsf{PUFSamp}}$. The changes between the two hybrids are only syntactical and therefore it holds that

$$\{EXC'_{\mathcal{H}_{2\cdot\ell(\lambda)+1},\mathcal{A},\mathcal{D}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}} = \{EXC'_{\mathcal{H}_{2\cdot\ell(\lambda)+2},\mathcal{A},\mathcal{D}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}}.$$

$\mathcal{H}_{2\cdot\ell(\lambda)+3}$: Let f be the following deterministic stateless oracle: f is initialized with the initial state of the physical token sent by Bob, the tuples $(k_1,\ldots,k_n)$, and a random tape. On input an index i, a set $\{q_j^0,q_j^1\}_{j\in[\ell(\lambda)]}$, and a set $\{p_j\}_{j\in[\ell(\lambda)]}$, the oracle f returns 1 if and only if for all $j\in[\ell(\lambda)]:\mathsf{hd}(q_j^{k_{i,j}},\beta_j)\leq\delta$, where β_j is the output of the token on input p_j. In this hybrid Alice no longer queries the token but computes the a valid opening for the i-th commitment only if f returns 1 on inputs i, $\{q_j^0,q_j^1\}_{j\in[\ell(\lambda)]}$, and $\{p_j\}_{j\in[\ell(\lambda)]}$. Where the elements $\{q_j^0,q_j^1\}_{j\in[\ell(\lambda)]}$ and $\{p_j\}_{j\in[\ell(\lambda)]}$ are defined in the i-th session. If f returns 0, then Alice interrupts all of the executions simultaneously. Note that this modification does not affect the view of the adversary: Since Alice keeps ownership of the token, the state of the token is not included in the longterm tapes. Also note that Alice never uses the values β_j, except for the check mentioned above. Thus we have that

$$\{EXC'_{\mathcal{H}_{2\cdot\ell(\lambda)+2},\mathcal{A},\mathcal{D}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}} = \{EXC'_{\mathcal{H}_{2\cdot\ell(\lambda)+3},\mathcal{A},\mathcal{D}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}}.$$

$\mathcal{H}_{2\cdot\ell(\lambda)+4}$: Let F_i be the set of tuples $\{q_j^0,q_j^1\}_{j\in[\ell(\lambda)]}$ and $\{p_j\}_{j\in[\ell(\lambda)]}$ such that f on input i and those tuples returns 0. Note that F_i is well defined as soon as Bob sends the token to Alice. In the i-th session, Alice no longer queries f but just checks whether $(\{q_j^0,q_j^1\}_{j\in[\ell(\lambda)]},\{p_j\}_{j\in[\ell(\lambda)]})\in\mathsf{F}_i$ and aborts all of the executions if this is the case. We denote by $\gamma\in\{1,\ldots,n,\infty\}$ the session in which Alice aborts. Here the two hybrids need to be equivalent only up to the first query to f that returns 0, thus

$$\{EXC'_{\mathcal{H}_{2\cdot\ell(\lambda)+3},\mathcal{A},\mathcal{D}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}} = \{EXC'_{\mathcal{H}_{2\cdot\ell(\lambda)+4},\mathcal{A},\mathcal{D}}(\lambda,z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}}.$$

$\mathcal{H}_{2\cdot\ell(\lambda)+5}$: Defined exactly as $\mathcal{H}_{2\cdot\ell(\lambda)+4}$ with the difference that for all sessions i of the protocol ω_i is computed as $\mathcal{H}_i\oplus m\|\mathsf{decom}$, where $\mathcal{H}_i$ is a random string in $\{0,1\}^c$. To prove the indistinguishability

of $\mathcal{H}_4$ and $\mathcal{H}_5$ we define the intermediate hybrids $(\mathcal{H}_{2\cdot\ell(\lambda)+4,0}, \ldots, \mathcal{H}_{2\cdot\ell(\lambda)+4,n})$, where in $\mathcal{H}_{2\cdot\ell(\lambda)+4,i}$ the strings $(\omega_1, \ldots, \omega_i)$ are computed as in $\mathcal{H}_{2\cdot\ell(\lambda)+5}$ whereas the strings $(\omega_{i+1}, \ldots, \omega_n)$ are computed as in $\mathcal{H}_{2\cdot\ell(\lambda)+4}$. Note that $\mathcal{H}_{2\cdot\ell(\lambda)+4,0} = \mathcal{H}_{2\cdot\ell(\lambda)+4}$ and $\mathcal{H}_{2\cdot\ell(\lambda)+4,n} = \mathcal{H}_{2\cdot\ell(\lambda)+5}$. By definition the hybrids $\mathcal{H}_{2\cdot\ell(\lambda)+4,i-1}$ and $\mathcal{H}_{2\cdot\ell(\lambda)+4,i}$ differ only in the value of $\mathcal{H}_i$ (which is $H(\mathsf{seed}, k_i)$ in the former case and a random string in the latter). Note that k_i is used only in the computation of $\mathcal{H}_i$ and that the only variable that depends on k_i is γ. Since γ is from a set of size $n+1$, we can bound from above the entropy loss of k_i to $log(n+1)$-many bits. Recall that $(n+1) \ll 2^\lambda$, therefore we have that $\ell(\lambda) - c > log(n+1)$, for an appropriate choice of $\ell()$. Hence, by the strong randomness of H we have that

$$\{EXC'_{\mathcal{H}_{2\cdot\ell(\lambda)+4,i-1},\mathcal{A},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}} \approx \{EXC'_{\mathcal{H}_{2\cdot\ell(\lambda)+4,i},\mathcal{A},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}}.$$

Since the distance between $\mathcal{H}_{2\cdot\ell(\lambda)+4,0}$ and $\mathcal{H}_{2\cdot\ell(\lambda)+4,n}$ is the sum of the bounds obtained by the leftover hash lemma [ILL89], we can conclude that

$$\{EXC'_{\mathcal{H}_{2\cdot\ell(\lambda)+4},\mathcal{A},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}} \approx \{EXC'_{\mathcal{H}_{2\cdot\ell(\lambda)+5},\mathcal{A},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}}.$$

$\mathcal{H}_{2\cdot\ell(\lambda)+6}$: Defined as $\mathcal{H}_{2\cdot\ell(\lambda)+5}$ except that in all sessions com is a commitment to a random string s. Note that in the execution of $\mathcal{H}_{2\cdot\ell(\lambda)+5}$ the value of decom is masked by a random string $\mathcal{H}_i$ and therefore it is information theoretically hidden to the eyes of the adversary. By the statistically hiding property of Com we have that for all unbounded distinguisher $\mathcal{A}$ the following holds:

$$\{EXC'_{\mathcal{H}_{2\cdot\ell(\lambda)+5},\mathcal{A},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}} \approx \{EXC'_{\mathcal{H}_{2\cdot\ell(\lambda)+6},\mathcal{A},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}}.$$

$\mathcal{H}_{2\cdot\ell(\lambda)+7}$: Defined as $\mathcal{H}_{2\cdot\ell(\lambda)+6}$ except that Alice opens the commitment to an arbitrary message m'. We observe that the execution of $\mathcal{H}_{2\cdot\ell(\lambda)+6}$ is completely independent from the message m, except when m is sent to Bob in clear in the opening phase. Therefore we have that for all unbounded distinguishers $\mathcal{D}$ that query the functionality polynomially-many times:

$$\{EXC'_{\mathcal{H}_{2\cdot\ell(\lambda)+6},\mathcal{A},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}} = \{EXC'_{\mathcal{H}_{2\cdot\ell(\lambda)+7},\mathcal{A},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}}.$$

S: We now define S as a simulator in the ideal world that engages the adversary in the simulation of a protocol when queried by the ideal functionality on input $(\mathsf{commited}, \mathsf{sid})$. The interaction of S with the adversary works exactly as specified in $\mathcal{H}_{2\cdot\ell(\lambda)+7}$, with the only difference that the message m' is set to be equal to x, where $(\mathsf{unveil}, \mathsf{sid}, x)$ is the message sent by the ideal functionality with the same value of sid. Since the simulation is unchanged to the eyes of the adversary we have that

$$\{EXC'_{\mathcal{H}_{2\cdot\ell(\lambda)+7},\mathcal{A},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}} = \{EXC'_{\rho,\mathsf{S},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}}.$$

By transitivity we have that $\mathcal{H}_0$ is statistically indistinguishable from S to the eyes of the environment $\mathcal{Z}$. We can conclude that our protocol everlastingly UC-realizes the commitment functionality $\mathcal{F}_{\mathsf{MCOM}}$ for any corrupted Bob. We stress that that we allow Bob to be computationally unbounded and we only require that the number of sessions is bounded by some polynomial in λ.

Corrupted Alice (committer). Let $\mathcal{H}_0$ be the execution of the protocol as described above in interaction with $\mathcal{A}$ and $\mathcal{Z}$. We define the following sequence of hybrids:

$\mathcal{H}_1$: Defined as $\mathcal{H}_0$ except that the the following algorithm is executed locally by Bob at the end of the commit phase of each session, in addition to Bob's normal actions.

$\mathcal{E}(1^\lambda)$: Let K be a bitstring of length $\ell(\lambda)$, the extractor parses the list of queries $\mathcal{Q}$ that Alice sent to $\mathcal{F}_{\mathsf{HToken}}^{\mathsf{PUFEval},\mathsf{PUFSamp}}$ before the last message of Bob in the commitment phase. Then for all $\mathcal{Q}_j \in \mathcal{Q}$ it checks whether $\exists j \in [\ell(\lambda)]$ such that $\exists z \in \{0,1\}$ such that $\mathsf{hd}(\mathcal{Q}_j, p_i^z) \leq \gamma$, where p_i^z is defined as in the original protocol. If this is the case the extractor sets $K_i = z$. If the value of K_i is already set to a different bit the extractor aborts. If at the end of list $\mathcal{Q}$ there is some i such that K_i is undefined, the extractor aborts. Otherwise it parses $\omega \oplus H(\mathsf{seed}, K)$ as $m'||\mathsf{decom}$ and it returns (m', decom).

Note that Bob does not use the output of $\mathcal{E}$ and therefore, for all distinguishers $\mathcal{D}$, we have that:

$$\{EXC'_{\mathcal{H}_0,\mathcal{A},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}} = \{EXC'_{\mathcal{H}_1,\mathcal{A},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}}.$$

$\mathcal{H}_2$: Let $\mathcal{H}_2$ be defined as $\mathcal{H}_1$ except that Bob outputs the message m' as computed by $\mathcal{E}$ instead of the message m as sent by Alice in the opening phase. For the indistinguishability of $\mathcal{H}_1$ and $\mathcal{H}_2$ we have to argue that if the opening of the adversary succeeds, then the extraction succeeds with overwhelming probability, i.e., $m = m'$. For the ease of exposition we assume that the sessions are enumerated with a unique identifier, e.g., according to their initialization order. Let Abort be the event such that there exists a session $j \in [n]$ such that the simulator aborts but the opening is successful. We are going to prove the following lemma.

Lemma 32. $\Pr[\mathsf{Abort} : \mathcal{H}_2] \leq \mathsf{negl}(\lambda)$.

Proof. We define $\mathsf{NoUnique}$ as the event such that there exists a session $j \in [n]$ such that the corresponding K^j as defined in $\mathcal{E}$ is not uniquely defined but the commitment is successful, i.e., there exists some $i \in [\ell(\lambda)]$ such that $K_i^j = 0$ and $K_i^j = 1$. Let $\mathsf{NoDefined}$ be the event such that there exists a $j \in [n]$ and $i \in [\ell(\lambda)]$ such that K_i^j is undefined at the end of the iteration, but the corresponding opening phase is successful. By definition of $\mathcal{E}$ we have that

$$\Pr[\mathsf{Abort} : \mathcal{H}_2] \leq \Pr[\mathsf{NoUnique} : \mathcal{H}_2] + \Pr[\mathsf{NoDefined} : \mathcal{H}_2].$$

The rest of the proof proceeds as follows:

- We show through a series of intermediate hybrids $(\mathcal{H}_0^{\mathsf{U}}, \ldots, \mathcal{H}_3^{\mathsf{U}})$ that the event $\mathsf{NoUnique}$ happens only with negligible probability.
- We show through a series of intermediate hybrids $(\mathcal{H}_0^{\mathsf{D}}, \ldots, \mathcal{H}_4^{\mathsf{D}})$ that the event $\mathsf{NoDefined}$ happens only with negligible probability.
- The proof of the lemma follows by a union bound.

We first derive a bound for the probability that the event $\mathsf{NoUnique}$ happens. Consider the following sequence of hybrids.

$\mathcal{H}_0^{\mathsf{U}}$: The experiment $\mathcal{H}_0^{\mathsf{U}}$ identical to $\mathcal{H}_2$ except that we sample some j^* from the identifiers associated to all sessions and some i^* from $[\ell(\lambda)]$. Let n be a bound on the total number of session and let $\mathsf{NoUnique}(j^*, i^*)$ be the event where $\mathsf{NoUnique}$ happens in session j^* and for the i^*-th bit. Since j^* and i^* are randomly chosen we have that

$$\Pr[\mathsf{NoUnique} : \mathcal{H}_2] \leq \Pr\left[\mathsf{NoUnique}(j^*, i^*) : \mathcal{H}_0^{\mathsf{U}}\right] \cdot n\ell(\lambda).$$

$\mathcal{H}_1^{\mathsf{U}}$: The experiment $\mathcal{H}_1^{\mathsf{U}}$ is defined as $\mathcal{H}_0^{\mathsf{U}}$ except that it stops before the execution of the i^*-th OT in session j^*. Let st be the state of all the machines in the execution of $\mathcal{H}_0^{\mathsf{U}}$, the experiment does the following:

- Continue the execution of $\mathcal{H}_0^{\mathsf{U}}$ from st.
- Input/output all the i^*-th OT messages from session j^*.
- Simulate all other messages internally.

The experiments sets the bit $b = 1$ if and only if the commitment of the j-th session succeeds. Let $\mathsf{NoUnique}^*(j^*, i^*)$ be the event that $K_{i^*}^{j^*}$ is not uniquely defined. Since the execution does not change to the eyes of Alice we have that

$$\Pr\left[\mathsf{NoUnique}(j^*, i^*) : \mathcal{H}_0^{\mathsf{U}}\right] = \Pr\left[b = 1 \wedge \mathsf{NoUnique}^*(j^*, i^*) : \mathcal{H}_1^{\mathsf{U}}\right].$$

$\mathcal{H}_2^{\mathsf{U}}$: Defined as $\mathcal{H}_2^{\mathsf{U}}$ except that the CRS for the OT is sampled to be in extraction mode. By the computational indistinguishability of the CRS, it holds that

$$\Pr\left[b = 1 \wedge \mathsf{NoUnique}^*(j^*, i^*) : \mathcal{H}_1^{\mathsf{U}}\right] \approx \Pr\left[b = 1 \wedge \mathsf{NoUnique}^*(j^*, i^*) : \mathcal{H}_2^{\mathsf{U}}\right].$$

$\mathcal{H}_3^{\mathsf{U}}$: Defined as $\mathcal{H}_2^{\mathsf{U}}$ except that the extractor for the OT is used in the i^*-th OT of the j^*-th session. The experiment sets $b = 1$ if the simulation succeeds. Recall that the simulator outputs the choice of the receiver b_{i^*} and expects as input the value $p_{i^*}^{b_{i^*}}$. Note that this implies that the value $p_{i^*}^{1-b_{i^*}}$ is information theoretically hidden to the eyes of Alice. Also note that

$$\begin{aligned}&\Pr\left[b = 1 \wedge \mathsf{NoUnique}^*(j^*, i^*) : \mathcal{H}_3^{\mathsf{U}}\right] = \\ &\textstyle\sum_{\mathsf{st}, i^*, j^*, p_{i^*}, \mathcal{Q}, \omega, \mathsf{seed}, \text{ s.t. } \mathsf{NoUnique}^*(j^*, i^*)} \cdot \Pr\left[b = 1 : \mathcal{H}_3^{\mathsf{U}}\right]\end{aligned}$$

by the simulation security of the OT we can rewrite

$$\begin{aligned}&\textstyle\sum_{\mathsf{st}, i^*, j^*, p_{i^*}, \mathcal{Q}, \omega, \mathsf{seed}, \text{ s.t. } \mathsf{NoUnique}^*(j^*, i^*)} \cdot \Pr\left[b = 1 : \mathcal{H}_3^{\mathsf{U}}\right] \geq \\ &\textstyle\sum_{\mathsf{st}, i^*, j^*, p_{i^*}, \mathcal{Q}, \omega, \mathsf{seed}, \text{ s.t. } \mathsf{NoUnique}^*(j^*, i^*)} \cdot \Pr\left[b = 1 : \mathcal{H}_2^{\mathsf{U}}\right]\end{aligned}$$

thus by Jensen's inequality we have that

$$\Pr\left[\mathsf{NoUnique}(j^*, i^*) : \mathcal{H}_3^{\mathsf{U}}\right] \geq \Pr\left[b = 1 \wedge \mathsf{NoUnique}^*(j^*, i^*) : \mathcal{H}_2^{\mathsf{U}}\right].$$

As we argued before the value of $p_{i^*}^{1-b_{i^*}}$ is information theoretically hidden to the eyes of Alice. However by definition of $\mathsf{NoUnique}^*(j^*, i^*)$ Alice queries both $(p_{i^*}^0, p_{i^*}^1)$ to the functionality $\mathcal{F}_{\mathsf{HToken}}^{\mathsf{PUFEval},\mathsf{PUFSamp}}$. It follows that we can bound the probability of the event $\mathsf{NoUnique}^*(j^*, i^*)$ to happen to a negligible function in the security parameter. Therefore we have that

$$\Pr[\mathsf{Abort} : \mathcal{H}_3] \leq \mathsf{negl}(\lambda) + \Pr[\mathsf{NoDefined} : \mathcal{H}_3].$$

In order to show a bound on the probability of $\mathsf{NoDefined}$ to happen in $\mathcal{H}_2$ we define another sequence of hybrids.

$\mathcal{H}_0^{\mathsf{D}}$: The experiment $\mathcal{H}_0^{\mathsf{D}}$ identical to $\mathcal{H}_2$ except that we sample some j^* from the identifiers associated to all sessions. Let n be a bound on the total number of session and let $\mathsf{NoDefined}(j^*)$ be the event where $\mathsf{NoDefined}$ happens for the session j. Since j^* is randomly chosen we have that

$$\Pr[\mathsf{NoDefined} : \mathcal{H}_2] \leq \Pr\left[\mathsf{NoDefined}(j^*) : \mathcal{H}_0^{\mathsf{D}}\right] \cdot n.$$

$\mathcal{H}_1^{\mathsf{D}}$: The experiment H_1^{B} is defined as H_0^{D} except that it stops before the execution of the SWIAoK in the commitment of session j^*. Let st be the state of all the machines in the execution of H_0^{D} under the assumption that no machine keeps a copy of the pre-image x after generating crs. Let $\mathcal{P}^*$ be the following algorithm:

- Continue the execution of H_0^{D} from st.
- Input/output all the SWIAoK messages from session j^*.
- Simulate all other messages internally.

The experiment H_1^{D} runs $b \leftarrow \left\langle \mathcal{P}^*(\mathsf{st}; r), \mathcal{V}_1(y, \{com_i\}_{i\in[\ell(\lambda)]}) \right\rangle$, where $\{com_i\}_{i\in[\ell(\lambda)]}$ are the messages sent in session j^* from Alice. Let $\mathsf{NoDefined}^*(j^*)$ be the event that there exists some $K_i^{j^*}$ that is undefined before the execution of the SWIAoK in session j^*. Then we have that

$$\Pr\left[\mathsf{NoDefined}(j^*) : H_0^{\mathsf{D}}\right] = \Pr\left[b = 1 \wedge \mathsf{NoDefined}^*(j^*) : H_1^{\mathsf{D}}\right],$$

by definition of $\mathsf{NoDefined}(j^*)$.

$\mathcal{H}_2^{\mathsf{D}}$: Defined as $\mathcal{H}_1^{\mathsf{D}}$ except that the extractor $(\{m_i, \mathsf{decom}_i\}_{i\in[\ell(\lambda)]}, x) \leftarrow \mathsf{Ext}^{\mathcal{P}^*(\mathsf{st})}(y, \{\mathsf{com}_i\}_{i\in[\ell(\lambda)]}; r)$ is executed instead of the SWIAoK. Note that

$$\Pr\left[\begin{array}{l} \{m_i, \mathsf{decom}_i\}_{i\in[\ell(\lambda)]} \leftarrow \mathsf{Ext}^{\mathcal{P}^*(\mathsf{st})}(y, \{\mathsf{com}_i\}_{i\in[\ell(\lambda)]}; r) : \\ (\{m_i, \mathsf{decom}_i\}_{i\in[\ell(\lambda)]}, (y, \{\mathsf{com}_i\}_{i\in[\ell(\lambda)]})) \in R_1 \\ \wedge \mathsf{NoDefined}^*(j^*) : \mathcal{H}_2^{\mathsf{D}} \end{array}\right] =$$
$$\sum_{\mathsf{st},\omega,\mathsf{seed},j^*,\mathcal{Q} \text{ s.t. } \mathsf{NoDefined}^*(j^*)} \Pr[\mathsf{st}, \omega, \mathsf{seed}, j^*, \mathcal{Q}]$$
$$\cdot \Pr\left[\begin{array}{l} \{m_i, \mathsf{decom}_i\}_{i\in[\ell(\lambda)]} \leftarrow \mathsf{Ext}^{\mathcal{P}^*(\mathsf{st})}(y, \{\mathsf{com}_i\}_{i\in[\ell(\lambda)]}; r) : \\ (\{m_i, \mathsf{decom}_i\}_{i\in[\ell(\lambda)]}, (y, \{\mathsf{com}_i\}_{i\in[\ell(\lambda)]})) \in R_1 \end{array}\right]$$

by the extraction property of the SWIAoK we can rewrite

$$\sum_{\mathsf{st},\omega,\mathsf{seed},j^*,\mathcal{Q} \text{ s.t. } \mathsf{NoDefined}^*(j^*)} \Pr[\mathsf{st}, \omega, \mathsf{seed}, j^*, \mathcal{Q}]$$
$$\cdot \Pr\left[\begin{array}{l} \{m_i, \mathsf{decom}_i\}_{i\in[\ell(\lambda)]} \leftarrow \mathsf{Ext}^{\mathcal{P}^*(\mathsf{st})}(y, \{\mathsf{com}_i\}_{i\in[\ell(\lambda)]}; r) : \\ (\{m_i, \mathsf{decom}_i\}_{i\in[\ell(\lambda)]}, (y, \{\mathsf{com}_i\}_{i\in[\ell(\lambda)]})) \in R_1 \end{array}\right] \geq$$
$$\sum_{\mathsf{st},\omega,\mathsf{seed},j^*,\mathcal{Q} \text{ s.t. } \mathsf{NoDefined}^*(j^*)} \Pr[\mathsf{st}, \omega, \mathsf{seed}, j^*, \mathcal{Q}]$$
$$\cdot \frac{\Pr\left[\ 1 = \left\langle \mathcal{P}^*(\mathsf{st}; r), \mathcal{V}_3(y, \{\mathsf{com}_i\}_{i\in[\ell(\lambda)]}) \right\rangle\ \right]^c}{p}.$$

By Jensen's inequality we can conclude that

$$\Pr\left[\begin{array}{l} \{m_i, \mathsf{decom}_i\}_{i\in[\ell(\lambda)]} \leftarrow \mathsf{Ext}^{\mathcal{P}^*(\mathsf{st})}(y, \{\mathsf{com}_i\}_{i\in[\ell(\lambda)]}; r) : \\ (\{m_i, \mathsf{decom}_i\}_{i\in[\ell(\lambda)]}, (y, \{\mathsf{com}_i\}_{i\in[\ell(\lambda)]})) \in R_1 \\ \wedge \mathsf{NoDefined}^*(j^*) : \mathcal{H}_2^{\mathsf{D}} \end{array}\right] \geq \frac{\Pr\left[b = 1 \wedge \mathsf{NoDefined}^*(j^*) : \mathcal{H}_1^{\mathsf{D}}\right]^c}{p}.$$

$\mathcal{H}_3^{\mathsf{D}}$: The experiment H_3^{B} is defined as H_2^{D} except that it stops before the execution of the SWIAoK in the opening of session j^*. Let st be the state of all the machines in the execution of H_2^{D} under the assumption that no machine keeps a copy of the trapdoor x after generating crs. Let $\mathcal{P}^*$ be the following algorithm:

- Continue the execution of H_2^{D} from st.
- Input/output all the SWIAoK messages from the opening phase of session j^*.
- Simulate all other messages internally.

The experiment H_1^{D} runs $b \leftarrow \left\langle \mathcal{P}^*(\mathsf{st}; r), \mathcal{V}_1(y, \mathsf{seed}, m, \mathsf{com}, \omega, \{com_i\}_{i\in[\ell(\lambda)]}, \{q_i^0, q_i^1\}_{i\in[\ell(\lambda)]}) \right\rangle$, where the input of the verification algorithm corresponds to the messages exchanged in session j^*.

To the eyes of Alice this change is only syntactical and therefore we have that

$$\Pr\left[\begin{array}{l}\{m_i,\mathsf{decom}_i\}_{i\in[\ell(\lambda)]}\leftarrow \mathsf{Ext}^{\mathcal{P}^*(\mathsf{st})}(y,\{\mathsf{com}_i\}_{i\in[\ell(\lambda)]};r):\\(\{m_i,\mathsf{decom}_i\}_{i\in[\ell(\lambda)]},(y,\{\mathsf{com}_i\}_{i\in[\ell(\lambda)]}))\in R_1\\\wedge\mathsf{NoDefined}^*(j^*):H_2^{\mathsf{D}}\end{array}\right]=$$
$$\Pr\left[\begin{array}{l}\{m_i,\mathsf{decom}_i\}_{i\in[\ell(\lambda)]}\leftarrow \mathsf{Ext}^{\mathcal{P}^*(\mathsf{st})}(y,\{\mathsf{com}_i\}_{i\in[\ell(\lambda)]};r):\\(\{m_i,\mathsf{decom}_i\}_{i\in[\ell(\lambda)]},(y,\{\mathsf{com}_i\}_{i\in[\ell(\lambda)]}))\in R_1\\\wedge b=1\wedge\mathsf{NoDefined}^*(j^*):H_3^{\mathsf{D}}\end{array}\right].$$

$\mathcal{H}_4^{\mathsf{D}}$: Defined as $\mathcal{H}_3^{\mathsf{D}}$ except that the extractor $(k,\{\mathsf{decom}_i\}_{i\in[\ell(\lambda)]},\mathsf{decom},x)\leftarrow\mathsf{Ext}^{\mathcal{P}^*(\mathsf{st})}(y,\mathsf{seed},m,\mathsf{com},\omega,\{com_i\}_{i\in[\ell(\lambda)]},\{q_i^0,q_i^1\}_{i\in[\ell(\lambda)]};r)$ is executed instead of the SWIAoK. An argument identical as above can be used to show that

$$\Pr\left[\begin{array}{l}\{m_i,\mathsf{decom}_i\}_{i\in[\ell(\lambda)]}\leftarrow \mathsf{Ext}^{\mathcal{P}^*(\mathsf{st})}(y,\{\mathsf{com}_i\}_{i\in[\ell(\lambda)]};r):\\(\{m_i,\mathsf{decom}_i\}_{i\in[\ell(\lambda)]},(y,\{\mathsf{com}_i\}_{i\in[\ell(\lambda)]}))\in R_1\wedge\\(k,\{\mathsf{decom}_i'\}_{i\in[\ell(\lambda)]},\mathsf{decom},x)\leftarrow\mathsf{Ext}^{\mathcal{P}^*(\mathsf{st})}\left(\begin{array}{l}y,\mathsf{seed},m,\mathsf{com},\omega,\\\{com_i,q_i^0,q_i^1\}_{i\in[\ell(\lambda)]}\end{array};r\right):\\((k,\{\mathsf{decom}_i'\}_{i\in[\ell(\lambda)]},\mathsf{decom},x),\left(\begin{array}{l}y,\mathsf{seed},m,\mathsf{com},\omega,\\\{com_i,q_i^0,q_i^1\}_{i\in[\ell(\lambda)]}\end{array}\right)\in R_2\\\wedge\mathsf{NoDefined}^*(j^*):\mathcal{H}_3^{\mathsf{D}}\end{array}\right]\geq$$
$$\underbrace{\Pr\left[\begin{array}{l}\{m_i,\mathsf{decom}_i\}_{i\in[\ell(\lambda)]}\leftarrow \mathsf{Ext}^{\mathcal{P}^*(\mathsf{st})}(y,\{\mathsf{com}_i\}_{i\in[\ell(\lambda)]};r):\\(\{m_i,\mathsf{decom}_i\}_{i\in[\ell(\lambda)]},(y,\{\mathsf{com}_i\}_{i\in[\ell(\lambda)]}))\in R_1\\\wedge b=1\wedge\mathsf{NoDefined}^*(j^*):\mathcal{H}_3^{\mathsf{D}}\end{array}\right]^c}_{p}.$$

In the following analysis we ignore the case where the two extracted witnesses are a valid trapdoor for the common reference string y, as this event can be easily ruled out with a reduction to the one-wayness of f. Let us denote by $\beta_i\leftarrow\mathsf{Open}(\mathsf{com}_i,\mathsf{decom}_i')$. Now it is now enough to observe that the successful termination of the protocol implies that for all $i\in[\ell(\lambda)]$ we have that $\mathsf{hd}(q_i^{k_i},\beta_i')\leq\delta$, for some $k=k_1||\ldots||k_\ell(\lambda)$. By definition of $\mathsf{NoDefined}^*(j^*)$ there exists some i^* such that $\mathcal{A}$ never queried any p' to $\mathcal{F}_{\mathsf{HToken}}^{\mathsf{PUFEval},\mathsf{PUFSamp}}$ such that neither $\mathsf{hd}(p',p_{i^*}^0)\leq\gamma$ nor $\mathsf{hd}(p',p_{i^*}^1)\leq\gamma$, before seeing the last message of the commitment phase. By the unpredictability of the PUF it follows that $\Pr\left[(\mathsf{hd}(m_{i^*},q_{i^*}^0)\leq\delta)\vee(\mathsf{hd}(m_{i^*},q_{i^*}^1)\leq\delta)\right]\leq\mathsf{negl}(\lambda)$. We can conclude that there exists an i^* such that $\beta_{i^*}\neq m_{i^*}$. Since decom_{i^*} and decom_{i^*}' are valid opening informations for m_{i^*} and β_{i^*}, respectively, then we can derive the following bound

$$\Pr\left[\begin{array}{l}\{m_i,\mathsf{decom}_i\}_{i\in[\ell(\lambda)]}\leftarrow \mathsf{Ext}^{\mathcal{P}^*(\mathsf{st})}(y,\{\mathsf{com}_i\}_{i\in[\ell(\lambda)]};r):\\(\{m_i,\mathsf{decom}_i\}_{i\in[\ell(\lambda)]},(y,\{\mathsf{com}_i\}_{i\in[\ell(\lambda)]}))\in R_1\wedge\\(k,\{\mathsf{decom}_i'\}_{i\in[\ell(\lambda)]},\mathsf{decom},x)\leftarrow\mathsf{Ext}^{\mathcal{P}^*(\mathsf{st})}\left(\begin{array}{l}y,\mathsf{seed},m,\mathsf{com},\omega,\\\{com_i,q_i^0,q_i^1\}_{i\in[\ell(\lambda)]}\end{array};r\right):\\((k,\{\mathsf{decom}_i'\}_{i\in[\ell(\lambda)]},\mathsf{decom},x),\left(\begin{array}{l}y,\mathsf{seed},m,\mathsf{com},\omega,\\\{com_i,q_i^0,q_i^1\}_{i\in[\ell(\lambda)]}\end{array}\right)\in R_2\\\wedge\mathsf{NoDefined}^*(j^*):\mathcal{H}_3^{\mathsf{D}}\end{array}\right]\leq\mathsf{negl}(\lambda),$$

by the binding property of the commitment scheme. Therefore we can conclude that

$$\Pr[\mathsf{Abort}:\mathcal{H}_2]\leq\mathsf{negl}(\lambda).$$

This proves our lemma. □

In order to conclude our proof we need to show that the extractor always returns a valid message-decommitment pair for the same message that Alice outputs in the opening phase. More formally, let NoExt be the event such that for the output of the extractor $(m', \mathsf{decom}) \leftarrow \mathcal{E}(1^\lambda)$ it holds that $m' \neq \mathsf{Open}(\mathsf{com}, \mathsf{decom})$, where com is the variable sent by Alice in the same session. Additionally, let BadExt be the event such that the output of extractor (m', decom) is a valid opening for com but $m' \neq m$, where m is the message sent by Alice in the opening for the same session. We are now going to argue that the probability that either NoExt or BadExt happens is bounded by a negligible function.

Lemma 33. $\Pr[\,\mathsf{NoExt} : \mathcal{H}_2] \leq \mathsf{negl}(\lambda)$.

Proof. Consider the sequence of games $\mathcal{H}_0^{\mathsf{D}}, \ldots, \mathcal{H}_4^{\mathsf{D}}$ as defined in the proof of Lemma 32. Let $\mathsf{NoExt}^*(j^*)$ be the event that the algorithm $\mathcal{E}$ returns an invalid opening for the commitment in session j^* and the extractors of the zero knowledge proofs output a valid pair of witnesses. With an argument along the same lines of the proof of Lemma 32 we can show that

$$\Pr[\,\mathsf{NoExt} : \mathcal{H}_2] \leq \frac{\Pr\left[\,\mathsf{NoExt}^*(j^*) : \mathcal{H}_4^{\mathsf{D}}\right]^c}{p}.$$

We now observe that whenever the extractor of the SWIAoK is successful then, for all $i \in [\ell(\lambda)]$ it holds that that $\beta_i \leftarrow \mathsf{Open}(\mathsf{com}_i, \mathsf{decom}'_i)$ and that $\mathsf{hd}(q_i^{k_i}, \beta_i) \leq \delta$, for some $k = k_1 || \ldots || k_{\ell(\lambda)}$. Additionally we have that $H(k, \mathsf{seed}) \oplus \omega$ is a valid decommitment information for com. By definition of NoExt we have that $m' \neq \mathsf{Open}(\mathsf{com}, \mathsf{decom})$, where (m', decom) is the output of $\mathcal{E}$ and it is defined as $\omega \oplus H(\mathsf{seed}, K)$. This implies that $K \neq k$, since the function H is deterministic. Therefore there must exists some i^* such that $K_{i^*} \neq k_{i^*}$. By Lemma 32 we know that K is uniquely defined and therefore Alice did not query $\mathcal{F}_{\mathsf{HToken}}^{\mathsf{PUFEval},\mathsf{PUFSamp}}$ for any p' such that $\mathsf{hd}(p_i^{z_i}, p') \leq \gamma$ for $z_i \neq K_i$, and therefore for all $i \in [\ell(\lambda)]$ it holds, by the unpredictability of the PUF, that

$$\Pr\left[\,\mathsf{hd}(\beta_i, q_i^{1-K_i}) \leq \delta\right] \leq \mathsf{negl}(\lambda),$$

and in particular we have that $\beta_{i^*} \neq m_{i^*}$. Since decom_{i^*} and decom'_{i^*} are valid openings for m_{i^*} and β_{i^*} with respect to com_{i^*}, the probability of $\mathsf{NoExt}^*(j^*)$ to happen in $\mathcal{H}_4^{\mathsf{D}}$ can be bound to a negligible function by the binding property of the commitment scheme. This proves the initial lemma. □

Lemma 34. $\Pr[\,\mathsf{BadExt} : \mathcal{H}_2] \leq \mathsf{negl}(\lambda)$.

Proof. The formal argument follows along the same lines as the proof of Lemma 33. The main observation here is that the argument implies that the output of $\mathcal{E}$ and the tuple (m, decom), where m is sent in plain by Alice and decom is the output of the extractor for the SWIAoK, must be identical with overwhelming probability. □

By the union bound we have that

$$\Pr[\mathsf{Abort} : \mathcal{H}_2] + \Pr[\mathsf{NoExt} : \mathcal{H}_2] + \Pr[\mathsf{BadExt} : \mathcal{H}_2] \leq \mathsf{negl}(\lambda).$$

It follows that for all session $j \in [n]$ our extractor as defined above does not abort except with negligible probability and outputs the same message that the adversary opens to with overwhelming probability. Therefore we can conclude that

$$\{EXC'_{\mathcal{H}_1,\mathcal{A},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}} \approx \{EXC'_{\mathcal{H}_2,\mathcal{A},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}}.$$

S: We can now define the simulator S that is identical to $\mathcal{H}_2$ except that the output m' of the algorithm $\mathcal{E}$ (defined as above) is used in the message $(\mathtt{commit}, \mathsf{sid}, m')$ to the ideal functionality $\mathcal{F}_{\mathsf{MCOM}}$. The corresponding decommitment message $(\mathtt{unveil}, \mathsf{sid})$ is sent when the adversary returns a valid decommitment to some message m. Since the interaction is unchanged to the eyes of the adversary, we have that

$$\{EXC'_{\mathcal{H}_2,\mathcal{A},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}} = \{EXC'_{\rho,\mathsf{S},\mathcal{D}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^{poly(\lambda)}}.$$

This implies that our protocol everlastingly UC-realizes the commitment functionality $\mathcal{F}_{\mathsf{MCOM}}$ for any corrupted Alice and concludes our proof. □

References

[AMSY16] Frederik Armknecht, Daisuke Moriyama, Ahmad-Reza Sadeghi, and Moti Yung. Towards a unified security model for physically unclonable functions. In Kazue Sako, editor, *Topics in Cryptology – CT-RSA 2016*, volume 9610 of *Lecture Notes in Computer Science*, pages 271–287, San Francisco, CA, USA, February 29 – March 4, 2016. Springer, Heidelberg, Germany.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th Annual ACM Symposium on Theory of Computing*, pages 479–488, Philadephia, PA, USA, May 22–24, 1996. ACM Press.

[BFSK11] Christina Brzuska, Marc Fischlin, Heike Schröder, and Stefan Katzenbeisser. Physically uncloneable functions in the universal composition framework. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 51–70, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.

[BKOV17] Saikrishna Badrinarayanan, Dakshita Khurana, Rafail Ostrovsky, and Ivan Visconti. Unconditional UC-secure computation with (stronger-malicious) PUFs. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 382–411, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press.

[CCM98] Christian Cachin, Claude Crépeau, and Julien Marcil. Oblivious transfer with a memory-bounded receiver. In *39th Annual Symposium on Foundations of Computer Science*, pages 493–502, Palo Alto, CA, USA, November 8–11, 1998. IEEE Computer Society Press.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[CGS08] Nishanth Chandran, Vipul Goyal, and Amit Sahai. New constructions for UC secure computation using tamper-proof hardware. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 545–562, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing*, pages 494–503, Montréal, Québec, Canada, May 19–21, 2002. ACM Press.

[CM97] Christian Cachin and Ueli M. Maurer. Unconditional security against memory-bounded adversaries. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 292–306, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Heidelberg, Germany.

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 418–430, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany.

[DFK+14] Dana Dachman-Soled, Nils Fleischhacker, Jonathan Katz, Anna Lysyanskaya, and Dominique Schröder. Feasibility and infeasibility of secure computation with malicious PUFs. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 405–420, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[DKMN15] Nico Döttling, Daniel Kraschewski, Jörn Müller-Quade, and Tobias Nilges. General statistically secure computation with bounded-resettable hardware tokens. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 319–344, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[DKMQ11] Nico Döttling, Daniel Kraschewski, and Jörn Müller-Quade. Unconditional and composable security using a single stateful tamper-proof hardware token. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 164–181, Providence, RI, USA, March 28–30, 2011. Springer, Heidelberg, Germany.

[DM08] Stefan Dziembowski and Ueli M. Maurer. The bare bounded-storage model: The tight bound on the storage requirement for key agreement. *IEEE Trans. Information Theory*, 54(6):2790–2792, 2008.

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 581–596, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany.

[DS13] Ivan Damgård and Alessandra Scafuro. Unconditionally secure and universally composable commitments from physical assumptions. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 100–119, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany.

[GIMS10] Vipul Goyal, Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. Interactive locking, zero-knowledge pcps, and unconditional cryptography. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 173–190, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.

[GIS+10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 308–326, Zurich, Switzerland, February 9–11, 2010. Springer, Heidelberg, Germany.

[HPV16] Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Composable security in the tamper-proof hardware model under minimal complexity. In *Theory of Cryptography Conference*, pages 367–399. Springer, 2016.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st Annual ACM Symposium on Theory of Computing*, pages 12–24, Seattle, WA, USA, May 15–17, 1989. ACM Press.

[Kat07] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 115–128, Barcelona, Spain, May 20–24, 2007. Springer, Heidelberg, Germany.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *20th Annual ACM Symposium on Theory of Computing*, pages 20–31, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[Mae13] Roel Maes. *Physically Unclonable Functions: Constructions, Properties and Applications*, volume 9783642413957. 11 2013.

[MMQN16] Jeremias Mechler, Jörn Müller-Quade, and Tobias Nilges. Universally composable (non-interactive) two-party computation from untrusted reusable hardware tokens. *IACR Cryptology ePrint Archive*, 2016:615, 2016.

[MQU07] Jörn Müller-Quade and Dominique Unruh. Long-term security and universal composability. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 41–60, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany.

[MQU10] Jörn Müller-Quade and Dominique Unruh. Long-term security and universal composability. *Journal of Cryptology*, 23(4):594–671, October 2010.

[MS08] Tal Moran and Gil Segev. David and Goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 527–544, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

[OOR+14] Claudio Orlandi, Rafail Ostrovsky, Vanishree Rao, Amit Sahai, and Ivan Visconti. Statistical concurrent non-malleable zero knowledge. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 167–191, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany.

[OSVW13] Rafail Ostrovsky, Alessandra Scafuro, Ivan Visconti, and Akshay Wadia. Universally composable secure computation with (malicious) physically uncloneable functions. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 702–718, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 187–196, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.

[Qua20] Willy Quach. Uc-secure OT from lwe, revisited. In Clemente Galdi and Vladimir Kolesnikov, editors, *Security and Cryptography for Networks - 12th International Conference, SCN 2020, Amalfi, Italy, September 14-16, 2020, Proceedings*, volume 12238 of *Lecture Notes in Computer Science*, pages 192–211. Springer, 2020.

[Rab03] Michael O. Rabin. Hyper encryption and everlasting secrets. In *Algorithms and Complexity, 5th Italian Conference, CIAC 2003, Rome, Italy, May 28-30, 2003, Proceedings*, pages 7–10, 2003.

[Unr13] Dominique Unruh. Everlasting multi-party computation. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 380–397, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.