

Composable Long-Term Security with Rewinding

Robin Berger¹, Brandon Broadnax², Michael Klooß¹, Jeremias Mechler¹,
Jörn Müller-Quade¹, Astrid Ottenhues¹, Markus Raiber¹

¹ KASTEL, Karlsruhe Institute of Technology, Karlsruhe, Germany
firstname.lastname@kit.edu

² broadnax@mail.informatik.kit.edu

Abstract. Long-term security, a variant of Universally Composable (UC) security introduced by Müller-Quade and Unruh (JoC '10), allows to analyze the security of protocols in a setting where all hardness assumptions no longer hold after the protocol execution has finished. Such a strict notion is highly desirable when properties such as input privacy need to be guaranteed for a long time, e.g. zero-knowledge proofs for secure electronic voting. Strong impossibility results rule out so-called *long-term-revealing* setups, e.g. a common reference string (CRS), to achieve long-term security, with known constructions for long-term security requiring hardware assumptions, e.g. signature cards. We circumvent these impossibility results by making use of new techniques, allowing rewinding-based simulation in a way that universal composability is possible. The new techniques allow us to construct a long-term-secure composable commitment scheme in the CRS-hybrid model, which is provably impossible in the notion of Müller-Quade and Unruh. We base our construction on a statistically hiding commitment scheme in the CRS-hybrid model with CCA-like properties. To provide a CCA oracle, we cannot rely on superpolynomial extraction techniques, as statistically hiding commitments do not define a unique value. Thus, we extract the value committed to via rewinding. However, even a CCA “rewinding oracle” without additional properties may be useless, as extracting a malicious committer could require to rewind other protocols the committer participates in. If this is e.g. a reduction, this clearly is forbidden. Fortunately, we can establish the well-known and important property of *k-robust extractability*, which guarantees that extraction is possible without rewinding *k*-round protocols the malicious committer participates in. While establishing this property for statistically binding commitment schemes is already non-trivial, it is even more complicated for statistically hiding ones. We then incorporate rewinding-based commitment extraction into the UC framework via a *helper* in analogy to Canetti, Lin and Pass (FOCS 2010), allowing both adversary and environment to extract statistically hiding commitments. Despite the rewinding, our variant of long-term security is universally composable. Our new framework provides the *first* setting in which a commitment scheme that is both statistically hiding and composable can be constructed from standard polynomial-time hardness assumptions and a CRS only. Unfortunately, we can prove that our setting does not admit long-term-secure oblivious transfer (and thus general two-party computations). Still, our long-term-secure commitment scheme suffices for natural applications, such as long-term secure and composable (commit-and-prove) zero-knowledge arguments of knowledge.

1 Introduction

Secure multi-party computation (MPC) allows mutually distrusting parties to perform computations on their private inputs, guaranteeing properties such as correctness, privacy or independence of inputs. To this end, the parties that wish to perform an MPC jointly execute a *protocol*, i.e. exchange messages with each other over a network.

In today’s highly connected world where devices are usually connected to the Internet, it is important to judge the security of cryptographic protocols not only in a *stand-alone setting* where only *one* instance of a protocol is executed at any time. Indeed, it is known that protocols which are stand-alone-secure may lose *all* security even if only two instances of *the same* protocol are executed in parallel [GK90].

In contrast, *composable security* considers security when multiple, possibly adversarially chosen, protocols may be executed concurrently. A very important notion for this setting is the so-called *Universally Composable (UC)* security, introduced by Canetti [Can01]. By default, UC security offers *computational* security only, i.e. crucially relies on hardness assumptions. If these hardness assumptions turn out to be wrong or eventually lose their validity (e.g. due to the availability of universal quantum computers), all security may be lost. For data that is very sensitive and must be kept secure for a long time, e.g. genomic data or electoral choices with secure online voting, computational (UC) security may be insufficient. In contrast, *statistical* security does not rely on hardness assumptions at all—offering, in principle, adequate security guarantees even for the most sensitive data. However, statistical (UC) security is often much harder to achieve than computational (UC) security, for example because a 2/3 honest majority [BGW88] or a very strong *trusted setup* [Bea97; DKM11] is needed.

Interestingly, there is a middle ground between statistical and computational security, called *long-term security*, introduced by Müller-Quade and Unruh [MU10]. With long-term security, hardness assumptions are only assumed to hold *during* the execution of a protocol. However, *after* the protocol execution has finished, all hardness assumptions become invalid and security must hold against (now) unbounded adversaries. Thus, long-term security offers a very interesting trade-off, in particular when considering that many protocol executions only take a comparatively short time compared to the assumed (and believed) validity of today’s used cryptographic hardness assumptions, which is often measured in years.

Impossibility Results of Long-Term Security. Unfortunately, long-term security is subject to very strong impossibility results: Many commonly used and natural setup assumptions (e.g. common reference strings (CRS) or certain public-key infrastructures (PKIs)) that suffice for UC security are too weak to construct long-term-secure composable commitment schemes. Indeed, the known constructions for long-term-secure and composable commitment schemes require very strong *hardware assumptions* such as (trusted) signature cards [MU10] or fully

malicious physically unclonable functions (PUFs) [Mag+22]. While these hardware assumptions are weaker than the assumptions required for statistical UC security, they are very impractical, as they require a great deal of coordination between protocol parties in order to exchange the necessary hardware.

Very informally, the strong impossibility results of long-term security come from the fact that commitments created by a malicious committer must be *extractable*, i.e. the simulator must be able to determine the value committed to. In a UC-secure commitment scheme that uses a CRS, this can be achieved by embedding an extraction trapdoor into the CRS (say, a public key used for encryption for which the secret decryption key is known). Using this extraction trapdoor, the value committed to can be determined in a *straight-line* way, i.e. without rewinding the malicious committer.

If long-term security is desired, the above approach fails: If a commitment scheme is straight-line extractable via an extraction trapdoor embedded into the CRS, the commitment must *statistically* contain the value committed to. When the protocol execution has finished, this extraction trapdoor can be (inefficiently) obtained and used to extract the commitment, contradicting the very requirement that the value committed to must be statistically hidden. Conversely, choosing the CRS distribution so that the commitment is statistically hiding makes straight-line extraction impossible. The example is generalized in [MU10], ruling out any “long-term revealing” setup, in particular *any* CRS distribution.

This problem, namely the insufficiency of widely-used and practical classes of setups such as common reference strings, raises our main research question:

Can we circumvent the impossibility results of [MU10] that rule out composable and long-term-secure commitment schemes from many natural setups?

Perhaps surprisingly, we can answer this question affirmatively and construct a long-term-secure composable commitment scheme in the CRS-hybrid model. In order to circumvent the prior impossibility result, we propose a new security notion based on (long-term) UC security which covers the possibility of extracting statistically hiding commitment schemes via rewinding.

Having circumvented the impossibility results of Müller-Quade and Unruh [MU10] for commitment schemes, we raise our second research question:

Can our techniques be used to construct protocols for composable general two-party computation with long-term security from natural setups?

We answer this question negatively by giving an impossibility result for oblivious transfer (OT) with long-term security for both sender and receiver. On the positive side, we sketch how OT with long-term security for *one* party can be achieved. This construction can then be generalized to (reactive) composable general two-party computation with long-term security for one party. Given our new impossibility result, this is the best one can hope for in the setting considered in this paper.

For the special case of (commit-and-prove) zero-knowledge, we give a positive result: Using our long-term-secure composable commitment scheme, we construct composable and long-term-secure (commit-and-prove) zero-knowledge.

1.1 Outline and Contribution

In this work, we construct protocols with composable long-term security for many important tasks. Our main contributions are as follows:

1. We introduce *pseudo-oracles* in Section 3.1. A pseudo-oracle \mathcal{O} is defined as an oracle which is given its caller's current view (and code), so that \mathcal{O} can rewind the caller in its head. With this, it is possible to extract a value from statistically hiding commitment schemes via rewinding.
2. As a building block for the long-term-secure commitment scheme, we construct a statistically hiding and equivocal CCA-secure commitment scheme in the CRS-hybrid model in Section 3. Here, we make use of pseudo-oracles to instantiate the CCA oracle.
3. We extend the notions of Universal Composability [Can01] and Long-Term Security [MU10] to a setting where commitments can be extracted by rewinding via a helper. As the helper is accessible by environment and adversary, we retain universal composability. We call the resulting security notion *Long-Term UC Security with Rewinding*.
4. We construct a composable commitment scheme which is long-term-Rewinding-UC-secure in the \mathcal{F}_{CRS} -hybrid model, circumventing the impossibility results of Müller-Quade and Unruh [MU10] in Section 6. To the best of our knowledge, we are the *first* to achieve such a strong notion of security for commitment schemes without resorting to hardware assumptions (or assumptions that already imply statistical UC security).
5. We present several applications of our commitment scheme in Section 7. Using our composable and long-term-secure commitment scheme, we can realize zero-knowledge as well as commit-and-proof with long-term security. Moreover, we obtain composable oblivious transfer with long-term security for one party. As we will show in Section 5, this is provably the best possible security in our setting.

Unless noted otherwise, we assume ideally authenticated communication and consider static corruptions only.

1.2 Technical Overview

As a first motivating example for the technical difficulty of this endeavor, first consider the possibility of giving *only the simulator* the ability to rewind the execution (as proposed by Nielsen [Nie03]). Conceptually, this approach is similar to e.g. security with superpolynomial simulation (SPS) [Pas03] where the simulator is given a (unilateral) complexity advantage. While such a model would allow the simulator to extract statistically hiding commitments, universal composability is lost due to the asymmetry between simulator and environment.

Universal Composability with (Superpolynomial) Helpers. In the case of SPS security, the problem of limited composability was solved by Prabhakaran and Sahai [PS04] and Canetti, Lin, and Pass [CLP10] by outsourcing the simulator’s additional capabilities to an external entity called *Imaginary Angel* [PS04] resp. *(superpolynomial) helper* [CLP10]. By making this entity available not only to the simulator but also to the environment, universal composability is recovered.

As we will use a similar approach to achieve composability, let us take a closer look at the approach taken by Canetti, Lin, and Pass [CLP10], that also illustrates the additional technical hurdles one faces when considering statistically hiding instead of statistically binding (composable) commitment schemes.

In [CLP10], the helper \mathcal{H} that provides the extraction oracle is parameterized with a fixed commitment scheme COM. Adversary and environment may, acting as committer for COM, perform commitments with \mathcal{H} , which acts as receiver. After the commit phase has finished, \mathcal{H} extracts the commitment with inefficient computations, solely based on the commitment transcript, which defines a value committed to with overwhelming probability. Here, no rewinding whatsoever is required. In order to prevent environment or adversary to create a commitment with \mathcal{H} as receiver in dependence on a commitment of an honest party, COM is assumed to feature the property of *security under chosen commitment attack (CCA)* as proposed in [CLP10]. Very informally, this property guarantees that an adversary cannot break the hiding property of a commitment it receives even if it has access to a commitment extraction oracle (for commitments where it plays the committer), subject to the condition that the *identity* (or *tag*) used in the commitment it receives (which should be hiding) is different from the identity used in the commitment it wants to have extracted through the committed-value oracle. Using COM, a composable commitment scheme is constructed.

For a statistically binding commitment scheme, the commitment transcript determines an unique value committed to with overwhelming probability, and that value can be recovered through inefficient computations. With statistically hiding commitments, this is not possible anymore. Merely requiring (CCA) extractability may not be very useful: For example, consider a UC protocol π that not only uses COM, but also a stand-alone-secure OT protocol π_{OT} . In order to reduce to the computational sender security of π_{OT} in the security proof of π , the reduction adversary needs to provide the UC environment and UC adversary with \mathcal{H} . In the reduction to a computational security property, this may not be possible if emulating \mathcal{H} requires super-polynomial computations (or, alternatively, requires rewinding the reduction). Interestingly, the CCA-secure commitment scheme constructed in [CLP10] features an additional property called *k-robustness*: For every PPT adversary \mathcal{A} interacting with the (inefficient) CCA oracle of COM and a k -round “external” protocol ϕ , there exists a PPT simulator interacting with ϕ , but *not* with the CCA oracle of COM, that has an output that is *computationally* indistinguishable from \mathcal{A} ’s output. The simulator \mathcal{S} internally emulates \mathcal{A} , but provides the CCA oracle by rewinding the internally executed \mathcal{A} in such a way that the external k -round protocol need *not* be rewound. This important property

enables the reuse of k -round UC-secure protocols in the first place, i.e. allows to show that \mathcal{H} does not negatively affect the protocols’ security.

Extracting Statistically Hiding Commitments Through Rewinding. In this work, we adapt the approach of Canetti, Lin, and Pass [CLP10] in such a way that extraction of statistically hiding commitments via *rewinding* is possible while universal composability is preserved. To this end, the extraction process must satisfy, informally, a) natural properties normally provided by (inefficient) committed-value oracles as well as b) robustness with respect to k -round protocols (as explained above).

Towards achieving these goals, we face a number of technical challenges that do not occur with “ordinary” inefficient oracles.

When the commitment scheme COM is not statistically *binding*, but statistically *hiding*, determining the value committed to is not possible from the commitment transcript. This follows from the very definition of the statistical hiding property. As such, the usual notion of a “committed-value oracle” is inapplicable. Instead, we show that it is possible to computationally define and extract the committed value via rewinding. To formalize this, we need to generalize the notion of oracles to capture rewinding.

To this end, we introduce the notion of *pseudo-oracles*, which are given their caller’s current view as an implicit input, thus allowing a pseudo-oracle to rewind its caller “in its head”.

Technical Challenges. In a composite execution of interactive algorithms \mathcal{B} and \mathcal{A} , denoted by $\langle \mathcal{B}, \mathcal{A} \rangle$, providing such a pseudo-oracle \mathcal{O} to \mathcal{A} may require to also provide \mathcal{B} ’s view to \mathcal{O} , because \mathcal{B} may be affected by the rewinding of \mathcal{A} performed by \mathcal{O} . Let $\langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}}$ denote this composed system. As outlined above, giving \mathcal{O} the view of \mathcal{B} may simply not be possible, e.g. because \mathcal{B} is the challenger in a security game. In contrast to normal oracles, pseudo-oracles do not generally feature the natural property of *composition-order invariance* (COI). Informally, this means that we may *not* simply consider $\langle \mathcal{B}, \mathcal{A}^{\mathcal{O}} \rangle$ (i.e. a system where \mathcal{O} only gets the view of \mathcal{A}) instead of $\langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}}$ (where \mathcal{O} gets the view of *both* \mathcal{B} and \mathcal{A}) and expect both systems to behave identically, despite the vastly different capabilities of \mathcal{O} in either case. Unlike with ordinary oracles that naturally satisfy COI (because they do not need access to their caller’s view), this essential property may simply not hold for arbitrary pseudo-oracles \mathcal{O} . If it does, because \mathcal{O} is specifically designed to do so, it requires a non-trivial proof.

Starting from the work of Goyal et al. [Goy+15], in particular the *robust extraction lemma* for PRS commitments [PRS02], we construct a statistically hiding commitment scheme CCACOM in the CRS-hybrid model that is rewinding-extractable, statistically trapdoor and features a committed-value pseudo-oracle that satisfies a notion of COI.³ Concretely, we define and prove the *k-robust*

³ The work [Goy+15] implicitly relies on pseudo-oracles, although it never explicitly acknowledges the fact that in the *general robust extraction lemma*, the “extractor” of a committed value is not an ordinary oracle anymore. We were unable to establish

composition-order invariance, which, informally, is COI for every k -round left side \mathcal{B} . Moreover, similar to the k -robustness property of [CLP10], we can replace a PPT adversary \mathcal{A} with access to our committed-value pseudo-oracle \mathcal{O} interacting with a k -round interactive algorithm \mathcal{B} (denoted by $\langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}}$) with a simulator \mathcal{S} *without* \mathcal{O} (denoted by $\langle \mathcal{B}, \mathcal{S} \rangle$), incurring a negligible *statistical* loss of security only. This property is particularly useful in reductions.

Looking ahead, these properties of \mathcal{O} will us allow to establish the a) statistical trapdoor property of CCACOM for adversaries with access to \mathcal{O} as well as the b) extractable-binding property of CCACOM for adversaries with access to \mathcal{O} , i.e. CCA-security-like properties for the commitment scheme CCACOM.

Universal Composability with Rewinding. Towards using pseudo-oracles for composable security, we modify the long-term (UC) execution to admit a helper that implement pseudo-oracles. To this end, we change the execution experiment to provide the pseudo-oracle with the necessary views. We call the resulting framework the *UC with Rewinding framework* (short *Rewinding UC framework*) and the resulting security notion *Long-Term UC Security with Rewinding* (short *Long-Term Rewinding UC security*). We also define a non-long-term variant of security in the presence of \mathcal{H} .

The notion of (long-term-) Rewinding UC security has very similar properties to the notion proposed by Canetti, Lin, and Pass [CLP10]: Not only is it closed under composition, but also is compatible with UC security for k -round protocols if the committed-value pseudo-oracle of the helper features k -robustness. This important property allows to import (computationally, long-term and statistically) UC-secure protocols into our framework.

Given that the helper may subtly affect the security of protocols, we provide several *justifications* for our new notion: We prove that (long-term-) Rewinding UC security implies standard (long-term) UC security resp. stand-alone real-ideal security for large classes of protocols. This demonstrates that our helper does not negatively affect the security of such protocols.

By importing our concurrently extractable and concurrently (statistically) trapdoor commitment scheme CCACOM into the Rewinding UC framework, we obtain a commitment scheme that satisfies the notion of long-term Rewinding UC security. To the best of our knowledge, this is the *first* construction in the CRS-hybrid model that is both composable and has long-term-like security. The construction can be instantiated from standard polynomial-time hardness assumptions and has an asymptotically optimal round complexity.

1.3 Related Work

Long-Term Security. In this paper, we build upon the notion of long-term security as introduced by Müller-Quade and Unruh [MU10]. In [MU10], a composable

the required properties of pseudo-oracles, in particular COI, for their schemes. While this does not mean that their schemes do not satisfy them, we stress that COI is an essential notion, which is non-trivial to establish. See Example 1 and Remark 14 for more discussions on such properties and Remark 15 for the relation to [Goy+15].

long-term secure commitment scheme is constructed from a trusted signature card. Conversely, large classes of setup assumptions, namely *long-term revealing* setups (Definition 21), have been shown to not admit long-term composable commitment schemes. In contrast to standard UC security [Can+02], long-term composable commitment schemes do not admit composable general multi-party computation with long-term security. Composable long-term commitment schemes also have previously been constructed from other hardware assumptions, namely fully malicious PUFs (together with a CRS) [Mag+22]. See Appendix B.3 for a short introduction to long-term security.

Statistical Security. Assuming the existence of ideally secure communication and an honest majority of more than $2/3$, composable general MPC with even statistical security is possible [AL17]. When there is no honest majority, strong setups such as tamper-proof hardware tokens (e.g. [DKM11]) also admit statistically secure composable general MPC. However, the use of such token-based protocols is often impractical, in particular when computations with many participants are desired. An alternative approach for statistical security is the use of pre-distributed correlated randomness [Bea97]. However, if the party providing the randomness is untrusted or becomes corrupted at any point, all security is lost.

Peikert, Vaikuntanathan, and Waters [PVW08] have presented several composable OT protocols. By using a CRS with appropriate distribution, statistical security for either the OT sender or the OT receiver is possible. However, if a CRS admitting statistical security for one party is chosen, composability is lost (see Appendix B.3 for a discussion). The same holds for many composable commitment schemes with statistical security for one party, e.g. [DN02].

Rewinding and Composable Security. With respect to concurrent self-composability, in particular for game-based security notions, several approaches using rewinding exist, e.g. for the case of commitment schemes [CLP10; Goy+15] or zero-knowledge [Kiy20; Orl+14]. To this end, a very helpful tool is a robust extraction lemma [Goy+15], which allows rewinding-based extraction without disturbing “left sides” up to a certain round complexity. While our protocol is based on [Goy+15] and their robust rewinding lemma, we modify it so that we can prove that the resulting pseudo-oracles satisfies essential properties, such as composition-order invariance (cf. Appendix C.1).

Allowing a UC simulator to rewind the execution has first been proposed by Nielsen [Nie03], leading to a security notion with properties reminiscent of SPS security, namely limited composability.

Canetti, Lin, and Pass [CLP10] have presented the first CCA-secure commitment scheme that can be extracted either straight-line by inefficient computations or efficiently using rewinding. Using this commitment scheme, they realize composable general MPC in the plain model, using the inefficient but straight-line extraction. Instead of allowing the simulator to perform these inefficient computations itself, they are performed by a special party called the *helper*. By also giving the environment access to the helper, the notion achieves universal composability. A drawback of the proposed approach is that UC reusability is limited. Intuitively,

this means that there exists a protocol π that UC-emulates a protocol ϕ , but that does not emulate ϕ under the new notion. However, for any polynomial k , the notion can be adapted such that any k -round UC-secure protocol can be reused, at the expense of a higher round complexity of the CCA-secure commitment scheme. This has subsequently improved, e.g. by [CLP13].

We use an approach that is reminiscent of the techniques of [CLP10]. In particular, we also use a helper to allow the extraction of commitments. As we are interested in commitment schemes with a statistical hiding property, straight-line extraction is not possible anymore. Instead, we let the helper rewind the execution, requiring changes to the execution experiment. For details, see Section 5.

2 Preliminaries

We use standard notation. The security parameter is denoted by κ . All inputs are assumed to be of length polynomial in κ . Often, we will provide Turing machines (usually the adversary, the environment as well as entities helping them) with input $(1^\kappa, z)$, where 1^κ denotes the unary encoding of κ and z is some (possibly non-uniform) input depending on κ . We use the usual notation for probability ensembles and write $\stackrel{c}{\approx}$ for computational and $\stackrel{s}{\approx}$ statistical indistinguishability.

For two interactive machines \mathcal{A} and \mathcal{B} , we write $\langle \mathcal{A}, \mathcal{B} \rangle$ to denote the system where \mathcal{A} and \mathcal{B} interact. We write $\text{out}_{\mathcal{B}} \langle \mathcal{A}(x), \mathcal{B}(y) \rangle (1^\kappa, z)$ (or similar) to denote the output of \mathcal{B} after interaction with \mathcal{A} , where \mathcal{A} and \mathcal{B} receive common input $(1^\kappa, z)$, and \mathcal{A} (resp. \mathcal{B}) receives private input x (resp. y). Similarly, we write $\text{view}_{\mathcal{A}}$ for the view of party \mathcal{A} , which consists of the party's random tape, all its inputs and all messages it received. By abuse of notation, we sometimes write $\langle \mathcal{A}, \mathcal{B} \rangle$ for a protocol. A k -round protocol sends at most k messages between its parties. An oracle algorithm \mathcal{A} has an "oracle interface" (i.e. expected input-output behavior), which can be filled in by an oracle \mathcal{O} (but also by a pseudo-oracle, cf. Section 3.1), and we write $\mathcal{A}^{\mathcal{O}}$ for the composed machine; an oracle \mathcal{O} is itself a (potentially unbounded) machine.

Two machines \mathcal{A}_0 and \mathcal{A}_1 are *indistinguishable w.r.t. rewinding*, if no unbounded distinguisher \mathcal{D} can distinguish black-box access to \mathcal{A}_0 or \mathcal{A}_1 .

2.1 Commitment Schemes

In our constructions, we use (stand-alone) commitment schemes as building blocks. In the following, we give a definition.

Definition 1 (Commitment Scheme). A commitment scheme (with setup), non-interactive unveil phase and message space \mathcal{M} is a tuple $(\text{Setup}, \mathbf{C}, \mathbf{R})$, with the following syntax: 1. **Setup** takes (1^κ) as input and outputs a commitment key ck , 2. \mathbf{C} and \mathbf{R} have common input $(1^\kappa, ck)$. We usually write (c, d) for the commitment-decommitment pair after the commit phase. The decommit phase is non-interactive and we write $\text{OPEN}(ck, c, v, d)$ for the function which outputs 1

if a decommitment d of commitment c to value v is accepted w.r.t. commitment key ck .

Definition 2 (Stateless and public-coin receivers). *A commitment scheme $\langle C, R \rangle$ is stateless, or more concretely, has a stateless receiver, if every message and output of the receiver is computed from the current (possibly empty) transcript. If the receiver’s messages are a (fixed) portion of its random tape (independent of the transcript), we call it a public-coin receiver.*

Many commitment schemes, in particular all non-interactive ones, are public-coin. Binding and hiding are defined as expected and omitted for space reasons. They are included in Appendix B.1 for completeness.

3 A Statistically Hiding Concurrently Extractable and Equivocal Commitment Scheme

In this section, we define pseudo-oracles which capture oracles which use rewinding. Building on this, we define CCA-security of commitment schemes w.r.t. a committed-value pseudo-oracle. Lastly, we construct a commitment scheme which is computationally CCA-binding and statistically CCA-equivocal (so in particular, statistically CCA-hiding). Our constructions and definitions are in the CRS model.

3.1 Pseudo-Oracles

For our protocols, we must offer the adversary access to a commitment-extraction *oracle*. However, the commitments are *statistically hiding*, so it is evidently impossible for any (unbounded) oracle to extract them. Thus, we relax the notion of “oracle-ness” to *pseudo-oracles*. The idea is, to make the view of the adversary accessible to the pseudo-oracle. Thus, it can execute and rewind it in its head. With this ability, even statistically hiding commitments can be extracted.

There is some freedom in the definition of pseudo-oracles and their properties. Our definition intentionally limits the power of pseudo-oracle as much as possible, see Remark 8 for a brief discussion. Fortunately, all our protocols should be robust to natural definitional changes to pseudo-oracles, so if more refined definitions are found, the protocols should remain their usefulness.

Definition 3 ((Pseudo-)Oracle). *Suppose \mathcal{A} is an (interactive) oracle algorithm. Suppose \mathcal{O} is a (stateful) algorithm which behaves like an oracle, i.e. interfaces with \mathcal{A} by responding to a query x with response y .*

- For (stateful) oracles, y is computed from \mathcal{O} ’s state, randomness, and query x .
- For (stateful) pseudo-oracles, y is computed from \mathcal{O} ’s state, randomness, and the current view of \mathcal{A} (which includes the query x to \mathcal{O}) as well as \mathcal{A} ’s code.

Remark 1 (Alternative interpretation). We give pseudo-oracles access to their caller’s code and view. Alternatively, we may restrict to *admissible callers*, which

pass their code and current view to the oracle (turning the pseudo-oracle into an ordinary oracle). Evidently, any oracle algorithm which uses a pseudo-oracle can be turned into an *admissible* oracle algorithm which uses an *ordinary* oracle.

The behavior of pseudo-oracles is quite different from ordinary oracles. For example, they allow to capture rewinding-based properties. Consequently, the familiar properties of ordinary oracles do not carry over in general, and we must make explicit the properties which a (pseudo-)oracle should have. In our setting, all pseudo-oracles of interest are *black-box*.

Definition 4 (Black-box Oracle). *A (possibly stateful) pseudo-oracle \mathcal{O} is black-box, if its output y is computed from \mathcal{O} 's state, randomness, black-box (rewinding) access to \mathcal{A} , and the current view of \mathcal{A} but with \mathcal{A} 's randomness removed (i.e. only all inputs and messages which \mathcal{A} received).*

Like ordinary oracles, black-box pseudo-oracles are independent of implementation details of their caller. We record following trivial consequence.

Corollary 1. *Let \mathcal{O} be a black-box pseudo-oracle. Let \mathcal{A} and \mathcal{B} be oracle algorithms which are perfectly indistinguishable w.r.t. rewinding. Then $\mathcal{A}^{\mathcal{O}}$ and $\mathcal{B}^{\mathcal{O}}$ are again perfectly indistinguishable w.r.t. rewinding.*

Another property of ordinary oracles is *composition-order invariance*, which asserts that, in a larger system of composed machines, it does not matter when a (pseudo-)oracle is connected to its caller.

Definition 5 (k -robust composition-order invariance (COI)). *A pseudo-oracle \mathcal{O} is k -robust composition-order invariant w.r.t. PPT algorithms, if for a pair of interacting PPT algorithms \mathcal{A} , \mathcal{B} , where $\langle \mathcal{B}, \mathcal{A} \rangle$ has at most k rounds, we have*

$$\begin{aligned} & \{\text{out}_{\mathcal{B}, \mathcal{A}}(\mathcal{B}(x), \mathcal{A}^{\mathcal{O}}(y))(1^\kappa, z)\}_{\kappa \in \mathbb{N}, x, y, z \in \{0,1\}^*} \\ & \stackrel{s}{\approx} \{\text{out}_{\mathcal{B}, \mathcal{A}}(\mathcal{B}(x), \mathcal{A}(y))^{\mathcal{O}}(1^\kappa, z)\}_{\kappa \in \mathbb{N}, x, y, z \in \{0,1\}^*}. \end{aligned}$$

That is, it is statistically indistinguishable whether the system was composed as

- $\langle \mathcal{B}, \mathcal{A}^{\mathcal{O}} \rangle$, *that is, first the pseudo-oracle \mathcal{O} is composed with \mathcal{A} , and then $\mathcal{A}^{\mathcal{O}}$ is composed with \mathcal{B} , or*
- $\langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}}$, *that is, first \mathcal{A} is composed with \mathcal{B} , and then the pseudo-oracle \mathcal{O} is composed with $\langle \mathcal{B}, \mathcal{A} \rangle$.*

We note that in the above, $\langle \mathcal{B}, \mathcal{A} \rangle$ is considered as a single entity, i.e. it is a single machine which emulates both \mathcal{B} , \mathcal{A} and their interaction. Consequently, for $\langle \mathcal{B}, \mathcal{A}^{\mathcal{O}} \rangle$, the pseudo-oracle has access to $\text{view}_{\mathcal{A}}$ only, whereas in $\langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}}$ it has access to $\text{view}_{\langle \mathcal{B}, \mathcal{A} \rangle}$ (where, by abuse of notation, we write $\text{view}_{\langle \mathcal{B}, \mathcal{A} \rangle}$ for the view of the entity $\langle \mathcal{B}, \mathcal{A} \rangle$ as explained above).

Definition 5 is quite abstract. It helps to consider the pseudo-oracle \mathcal{O}_{CCA} from Section 3.2. There, the core difference between $\langle \mathcal{B}, \mathcal{A}^{\mathcal{O}} \rangle$ and $\langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}}$ is, whether it is possible to rewind \mathcal{B} alongside \mathcal{A} , or not (because \mathcal{B} is an external entity). Composition-order invariance for \mathcal{O}_{CCA} intuitively ensures that, despite this difference, the values extracted by \mathcal{O}_{CCA} for \mathcal{A} remain unchanged.

Remark 2 (Relation to oracles). Ordinary oracles are evidently black-box and ∞ -robust composition-order invariant (w.r.t. to unbounded algorithms).

We stress that composition-order invariance is a non-trivial property of pseudo-oracles. Indeed, to verify that the CCA-commitment pseudo-oracle \mathcal{O}_{CCA} satisfies composition-order invariance, we crucially rely on computational assumptions; and we do not know whether it holds without such assumptions. For that reason, Definition 5 restricts to PPT algorithms.

Another useful property allows the elimination of a pseudo-oracle altogether. This corresponds to the k -robustness property of [CLP10; Goy+15].

Definition 6 (k -robust pseudo-PPT). *A black-box pseudo-oracle is k -robust pseudo-PPT if for every (interactive) PPT oracle algorithm \mathcal{A} , there exists a PPT algorithm \mathcal{S} such that for every interactive PPT algorithm \mathcal{B} interacting with \mathcal{A} in at most k rounds, we have*

$$\begin{aligned} & \{\text{out}_{\mathcal{B},\mathcal{A}}(\mathcal{B}(x), \mathcal{A}^\mathcal{O}(y))(1^\kappa, z)\}_{\kappa \in \mathbb{N}, x, y, z \in \{0,1\}^*} \\ & \stackrel{s}{\approx} \{\text{out}_{\mathcal{B},\mathcal{S}}(\mathcal{B}(x), \mathcal{S}(y))(1^\kappa, z)\}_{\kappa \in \mathbb{N}, x, y, z \in \{0,1\}^*}. \end{aligned}$$

Besides these generic properties of pseudo-oracles, more concrete properties are of interest in many cases. For example, for our \mathcal{O}_{CCA} pseudo-oracle we will prove a “substitution rule” in Appendix C.2. This continues the pattern, that properties which are obvious for oracles need not be obvious for pseudo-oracles, but can (and must) be explicitly established.

3.2 Properties of Commitment Schemes

We require a commitment scheme which is *concurrently extractable* and *statistically hiding* (indeed, equivocal) to achieve concurrent security. Consequently, we define security notions in the presence of a committed-value oracle \mathcal{O}_{CCA} . The security w.r.t. a committed-value oracle will be important, as it allows to concurrently extract adversarial inputs to commitments, while at the same time simulating commitments of honest parties. The committed-value oracle \mathcal{O}_{CCA} for COM plays the receiver of COM in an arbitrary number of sessions. Upon completion of a commitment phase in session s , \mathcal{O}_{CCA} outputs $(\text{End}, s, v_s, \text{view}_{R_s})$. (The view of the receiver is outputted for technical reasons. For public-coin COM, it contains no additional information anyway.)

Definition 7 (CCA-binding). *Let COM be a commitment scheme with message space \mathcal{M} . Let \mathcal{O}_{CCA} be a pseudo-oracle whose interface is described below. Let $\text{Exp}_{\mathcal{A}, \mathcal{O}_{\text{CCA}}, \text{COM}}^{\text{CCA-bind}}(\kappa, z)$ be the output of the following experiment:*

1. *Run the adversary \mathcal{A} on input $(1^\kappa, z)$ with access to a (committed-value) pseudo-oracle \mathcal{O}_{CCA} provided through the game \mathcal{G} . Formally, this means $\langle \mathcal{G}, \mathcal{A} \rangle^{\mathcal{O}_{\text{CCA}}}$ where game \mathcal{G} passes messages between \mathcal{A} and \mathcal{O}_{CCA} as follows:*
 - \mathcal{O}_{CCA} is proxied through \mathcal{G} and allows \mathcal{A} to choose common inputs $(1^\kappa, t)$ and interact with an honest receiver R_s in session s (for arbitrarily many concurrent sessions). For this, \mathcal{O}_{CCA} first generates a fresh setup $\text{ck}_s \leftarrow \text{Setup}(1^\kappa, t)$ and sends it to \mathcal{A} (through \mathcal{G}).

- Whenever a session s is finished, \mathcal{O}_{CCA} responds with $(\text{End}, s, v_s, \text{view}_{R_s})$, where v_s is the extracted value of commitment (which may be \perp , e.g. if the receiver does not accept) or \perp_{ext} if extraction failed. The game passes (End, s, v_s) to \mathcal{A} (but not view_{R_s}).
- 2. In any session s whose commit phase finished, the adversary may complete the unveil phase for s . This phase is simulated by the game (which has access to view_{R_s}). Suppose receiver R_s accepts opening to $v \neq \perp$. If $v \neq v_s$ and $v \in \mathcal{M}$, then the game outputs 1, i.e. \mathcal{A} wins. (For $v = \perp$, \mathcal{A} does not win.)
- 3. If \mathcal{A} stops, the game outputs 0, i.e. \mathcal{A} loses the game.

We say COM is CCA-binding w.r.t. \mathcal{O}_{CCA} , if for every PPT adversary \mathcal{A} , there exists a negligible function negl , such that for all $\kappa \in \mathbb{N}$ and all $z \in \{0, 1\}^*$, $\Pr[\text{Exp}_{\mathcal{A}, \mathcal{O}_{\text{CCA}}, \text{COM}}^{\text{CCA-bind}}(\kappa, z) = 1] \leq \text{negl}$. We then call \mathcal{O}_{CCA} a committed-value pseudo-oracle.

Some remarks are in order: Firstly, Definition 7 is the binding analogue to CCA-hiding of [CLP10]. Secondly, it is a multi-challenge variant (but as usual, multi- and single-challenge are equivalent by a standard argument). Thirdly, it would have been more straightforward to have the game or \mathcal{O}_{CCA} play the honest receivers in both commit and unveil phase. However, we want a committed-value pseudo-oracle with the interface as in Definition 7, i.e. only the commitment phase. Lastly, \mathcal{O}_{CCA} even rewinds the game, but this only strengthening the notion of binding. We note that, if \mathcal{O}_{CCA} is $O(1)$ -robust COI, then a game with a constant number of challenge sessions can be handled as a left side, so that \mathcal{O}_{CCA} does not rewind the game anymore.

Terminology 1 (Value Committed To) Let COM be a commitment scheme which is CCA-binding w.r.t. committed-value oracle \mathcal{O}_{CCA} . Let $v \in \mathcal{M} \cup \{\perp, \perp_{\text{ext}}\}$ be the value which is part of the output of \mathcal{O}_{CCA} in an interaction with a (possibly malicious) committer. If $v \in \mathcal{M}$, we say that v is the value committed to. If $v \notin \mathcal{M}$, i.e. $v \in \{\perp, \perp_{\text{ext}}\}$, we do not consider the commitment to have a value.

For straight-line simulation of commitments, we define trapdoor commitment schemes. These allow to generate and equivocate dummy commitments.

Definition 8 (Trapdoor Commitment Scheme). Let (Setup, C, R) be a commitment scheme with message space \mathcal{M} , and let $(\text{TSetup}, C_{\text{trap}})$ be algorithms which can be used in place of (Setup, C) . Let \mathcal{O}_{CCA} be a committed-value pseudo-oracle. Then $\text{TRAPCOM} = (\text{Setup}, C, R, \text{TSetup}, C_{\text{trap}})$ is called trapdoor w.r.t. \mathcal{O}_{CCA} if

- $\langle C, R \rangle$ and $\langle C_{\text{trap}}, R \rangle$ are commitment schemes with message space \mathcal{M} , and
- for all PPT adversaries \mathcal{A} , it holds that

$$\begin{aligned} & \{\text{Exp}_{\mathcal{A}, \mathcal{O}_{\text{CCA}}, \text{TRAPCOM}}^{\text{TDC}}(\kappa, 0, z)\}_{\kappa \in \mathbb{N}, z \in \{0, 1\}^*} \\ & \stackrel{s}{\approx} \{\text{Exp}_{\mathcal{A}, \mathcal{O}_{\text{CCA}}, \text{TRAPCOM}}^{\text{TDC}}(\kappa, 1, z)\}_{\kappa \in \mathbb{N}, z \in \{0, 1\}^*} \end{aligned}$$

that is, the ensembles are statistically indistinguishable.

The experiment $\text{Exp}_{\mathcal{A}, \mathcal{O}_{\text{CCA}}, \text{TRAPCOM}}^{\text{TDC}}(\kappa, b, z)$ is defined as follows:

1. Run $\mathcal{A}(1^\kappa, z)$ where \mathcal{A} interacts with the game \mathcal{G} as follows.
2. First, \mathcal{A} sends (**Setup**). If $b = 0$, set $\text{ck} \leftarrow \text{Setup}(1^\kappa)$. Otherwise, set $(\text{ck}, \text{td}) \leftarrow \text{TSetup}(1^\kappa)$. The experiment sends ck to \mathcal{A} .
3. By sending (**Start**, v) to \mathcal{G} , \mathcal{A} starts the commit phase of TRAPCOM, acting as receiver. If $b = 0$, the experiment runs the code of the honest committer \mathcal{C} on input $(1^\kappa, \text{ck}, v)$. If $b = 1$, the experiment runs the code of the trapdoor committer $\mathcal{C}_{\text{trap}}$ on input $(1^\kappa, \text{ck}, |v|, \text{td})$.
4. After the commit phase has finished, wait for a message (**Unveil**) and perform the unveil phase. If $b = 1$, the trapdoor committer receives v as additional private input.
5. The experiment gives \mathcal{A} access to a committed-value pseudo-oracle \mathcal{O}_{CCA} as in Definition 7. Concretely, we consider $\langle \mathcal{G}, \mathcal{A} \rangle^{\mathcal{O}_{\text{CCA}}}$ as the complete experiment.
6. The experiment outputs the view of \mathcal{A} .

As for CCA-binding, in the CCA-equivocation experiment, \mathcal{O}_{CCA} rewinds the whole game. This only makes the adversary (and hence security notion) stronger. Unlike CCA-binding, Definition 8 is single-challenge.

3.3 Constructions

We first recall the definition of a PRS commitment for μ -bit messages from [Goy+15].

Construction 1 (PRS Commitment Scheme (adapted from [Goy+15]))

Let $\kappa \in \mathbb{N}$ be a security parameter. Let $\text{COM}' = (\text{Setup}', \mathcal{C}', \mathcal{R}')$ be a commitment scheme with message space $\mathcal{M} = \{0, 1\}^{\mu(\kappa)}$ for polynomial μ . Let $\ell = \ell(\kappa)$ denote a round parameter. The PRS commitment scheme with ℓ rounds and base commitment COM' is denoted PRS_ℓ or just PRS. It has message space \mathcal{M} and is defined as follows.

Setup. Generate $\text{ck}_{i,j}^b \leftarrow \text{Setup}'(1^\kappa)$ for $b \in \{0, 1\}$, $i \in [\kappa]$, $j \in [\ell]$.

Commit Phase. On common input (1^κ) and private input $v \in \mathcal{M}$ for \mathcal{C} :

1. The committer \mathcal{C} chooses $\kappa \cdot \ell$ pairs of random shares $(s_{i,j}^0, s_{i,j}^1)$ of v , i.e. for all i, j it holds that $s_{i,j}^0 \oplus s_{i,j}^1 = v$. Then, \mathcal{C} runs \mathcal{C}' to commit to $s_{i,j}^b$ for $b \in \{0, 1\}$, $i \in [\kappa]$, $j \in [\ell]$ under commitment key $\text{ck}_{i,j}^b$ (in parallel) to obtain commitments $c_{i,j}^b$.
2. For $j = 1, \dots, \ell$ sequentially:
 - (a) The receiver \mathcal{R} sends a challenge string $r_j = (r_{1,j}, \dots, r_{\kappa,j}) \xleftarrow{\$} \{0, 1\}^\kappa$.
 - (b) The committer \mathcal{C} unveils the commitments $c_{1,j}^{r_{1,j}}, \dots, c_{\kappa,j}^{r_{\kappa,j}}$. The receiver aborts if any unveil is invalid.

Unveil Phase. 1. The committer unveils all remaining shares that have not been opened in the commit phase.

2. The receiver accepts a value v if $u_{1,1}^0 \oplus u_{1,1}^1 = \dots = u_{\kappa,\ell}^0 \oplus u_{\kappa,\ell}^1 = v$, where $u_{i,j}^b$ is the message unveiled for commitment $c_{i,j}^b$.

Terminology 2 In the following, we call the commitment schemes used within the PRS commitment scheme elementary or base commitment schemes, to distinguish them from the PRS commitment scheme itself.

Theorem 1. Let $k \in \mathbb{N}$ be a parameter (which may depend on κ). Let COM' be a base commitment scheme such that:

1. COM' has a public-coin receiver and non-interactive unveil phase.
 2. The commitment phase of COM' has at most k rounds and the first message is sent by the committer C_t .
 3. COM' is binding.
 4. COM' is trapdoor with trapdoor committing algorithm C'_{trap} .
- Define a commitment scheme CCACOM and round parameter $\ell \in \omega(k(\kappa) \log(\kappa))$ as follows:

- **Inputs:** Common input is (1^κ) . Private input to C is v .
- **Setup:** Setup for PRS_ℓ (i.e. for base commitment COM' and ℓ rounds).
- **Commit Phase:**
 1. **PRS commit:** Run the PRS commitment phase of PRS_ℓ . Let τ_{prs} be the PRS commitment transcript.
 2. **AoK:** Run Blum’s graph hamiltonicity protocol κ -fold in parallel with base commitments COM' to prove: τ_{prs} is a valid PRS commitment to some value $v \in \mathcal{M}$.
- **Unveil Phase:** Run the corresponding PRS unveil phase.

Let \mathcal{O}_{CCA} be the following pseudo-oracle:

- \mathcal{O}_{CCA} allows \mathcal{A} to choose common inputs (1^κ) and interact with an honest receiver R_s in session s in arbitrarily many concurrent sessions. For this, \mathcal{O}_{CCA} first generates a fresh setup $\text{ck}_s \leftarrow \text{Setup}(1^\kappa)$ and sends it to \mathcal{A} .
- \mathcal{O}_{CCA} runs the rewinding-based extraction of PRS commitments as in [Goy+15], c.f. Appendix C.1 for more details. Let v_s denote the extracted value (which may be \perp) received in (main thread) session s . (If extraction failed, v_s is the special symbol \perp_{ext} .)
- When the commitment phase of session s completes, \mathcal{O}_{CCA} outputs $(\text{End}, s, v_s, \text{view}_{R_s})$ where v_s is replaced by \perp if R rejected the AoK.

Then the following holds for CCACOM w.r.t. \mathcal{O}_{CCA} :

1. CCACOM has at most $O(k \cdot (\ell + 1))$ rounds and non-interactive unveil phase.
2. CCACOM is CCA-binding w.r.t. \mathcal{O}_{CCA} .
3. CCACOM is trapdoor for some trapdoor committing algorithm C'_{trap} .
4. \mathcal{O}_{CCA} is black-box and $O(k)$ -robust composition-order invariant.
5. \mathcal{O}_{CCA} is $O(k)$ -robust pseudo-PPT.

Note that in Theorem 1, we use a parallel repetition of Blum’s graph hamiltonicity protocol instead of a general AoK. This is primarily for simplicity.

Proof (Proof sketch). The claims in Item 1 follow immediately. Item 2, i.e. CCA-binding w.r.t. \mathcal{O}_{CCA} , follows immediately from Lemma 3, the generalized robust extraction lemma of [Goy+15] (adapted to our setting). The rewinding schedule of [Goy+15] only uses black-box rewinding access and at most squares the worst-case runtime, hence Item 5, i.e. black-boxness and pseudo-PPT, follow as well. It remains to show Item 3 and Item 4.

Claim 1 (Item 4). \mathcal{O}_{CCA} is $O(k)$ -robust composition-order invariant.

We defer the proof of this claim to Section 4.2.

Finally, we show the trapdoor property, Item 3. The trapdoor commitment setup is obtained by (in the PRS commitment) replacing Setup' with TSetup' and C' with C'_{trap} , i.e., simply using the trapdoor commitment algorithms of the base commitment COM' .

Claim 2 (Item 3). COM is trapdoor w.r.t. \mathcal{O}_{CCA} for trapdoor commitment algorithms $(\text{TSetup}, C_{\text{trap}})$ as described above.

To prove indistinguishability, we use following hybrids.

- Hybrid H_0 : Real game, i.e. the bit b in $\text{Exp}_{\mathcal{A}, \mathcal{O}_{\text{CCA}}, \text{TRAPCOM}}^{\text{TDC}}$ is 0.
- Hybrid H_j : Same as H_0 , except that for PRS slots $j' \leq j$, the hybrid executes the left commit phase using trapdoor algorithms TSetup' and C'_{trap} (and trapdoor unveil) instead of the real algorithms.
- Hybrid H_ℓ : The simulation, i.e. the bit b in $\text{Exp}_{\mathcal{A}, \mathcal{O}_{\text{CCA}}, \text{TRAPCOM}}^{\text{TDC}}$ is 1.

All hybrids output the view of \mathcal{A} , denoted by $\text{view}_{\mathcal{A}}$. Intuitively, indistinguishability of H_i and H_{i+1} obviously follows from trapdoor property Definition 8. However, the presence of rewinding \mathcal{O}_{CCA} makes the argument non-trivial and we have to rely on COI. We sketch how the formal argument and embedding works. Let G_i be the experiment H_i with \mathcal{O}_{CCA} factored out, i.e. $H_i = G_i^{\mathcal{O}_{\text{CCA}}}$. Let $E_b := \text{Exp}_{\mathcal{H}_i, \mathcal{O}_{\text{CCA}}, \text{TRAPCOM}'^{2\kappa}}^{\text{TDC}}(1^\kappa, b, z)$ be the TDC experiment from Definition 8. Finally, let \mathcal{G}_i be defined such that $G_i = \langle E_0, \mathcal{G}_i \rangle$, i.e. make session i explicit and “move” it into E_b . Also observe that $G_{i+1} = \langle E_1, \mathcal{G}_i \rangle$. This gives us

$$H_i = G_i^{\mathcal{O}_{\text{CCA}}} = \text{out}_{\mathcal{G}_i} \langle E_0, \mathcal{G}_i \rangle^{\mathcal{O}_{\text{CCA}}} \stackrel{s}{\approx} \text{out}_{\mathcal{G}_i} \langle E_0, \mathcal{G}_i^{\mathcal{O}_{\text{CCA}}} \rangle,$$

where the first equality holds by definition, the next equality follows by *black-boxness* of \mathcal{O}_{CCA} (Corollary 1), and the statistical indistinguishability follows from $O(k)$ -robust COI of \mathcal{O}_{CCA} (for PPT algorithms). We stress that this indistinguishability is *not* automatic for pseudo-oracles and *requires justification*, given by COI. Next, we find

$$\text{out}_{\mathcal{G}_i} \langle E_0, \mathcal{G}_i^{\mathcal{O}_{\text{CCA}}} \rangle \stackrel{s}{\approx} \text{out}_{\mathcal{G}_i} \langle E_1, \mathcal{G}_i^{\mathcal{O}_{\text{CCA}}} \rangle,$$

by using that COM' is a trapdoor commitment by assumption. Now, we reverse the previous steps, with the same arguments but E_1 as left side to find

$$\text{out}_{\mathcal{G}_i} \langle E_1, \mathcal{G}_i^{\mathcal{O}_{\text{CCA}}} \rangle \stackrel{s}{\approx} \text{out}_{\mathcal{G}_i} \langle E_1, \mathcal{G}_i \rangle^{\mathcal{O}_{\text{CCA}}} = G_{i+1}^{\mathcal{O}_{\text{CCA}}} = H_{i+1}.$$

Thus, we have shown that $H_i \stackrel{s}{\approx} H_{i+1}$ as claimed. This finishes the proof. \square

The base commitment COM' in Theorem 1 can be instantiated using a number of assumptions.

Proposition 1 (Possible Instantiations). *Under the RSA assumption, the DLOG assumption and the SIS assumption, there exist commitment schemes COM'_{RSA} [HW09], $\text{COM}'_{\text{DLOG}}$ [Ped92] and COM'_{SIS} [GVW15] in the CRS-hybrid model with*

1. a public-coin receiver and non-interactive unveil phase,
2. a commit phase of $O(1)$ rounds and the first message is sent by the committer,
3. a computational binding property and
4. a statistical trapdoor property.

4 Analysis of the Committed-Value Oracle \mathcal{O}_{CCA}

In this section, we recall the PRS rewinding schedule from [Goy+15], presented in our setting. Moreover, we show that the committed-value oracle \mathcal{O}_{CCA} from Theorem 1 satisfies composition-order invariance. In Appendix C, we recall the robust extraction lemma from [Goy+15] and present a useful substitution rule for \mathcal{O}_{CCA} , namely the possibility to move receiver session in and out of \mathcal{O}_{CCA} (so that they can be implemented and used in a security reduction).

4.1 The Rewinding Schedule from [Goy+15]

We recall the rewinding schedule of [Goy+15], which itself is based on [PRS02; PTV14]. In [Goy+15], an adversary \mathcal{A} interacting with an external party \mathcal{B} and an (external) PRS receiver is considered. Thus, \mathcal{A} can send messages to

- the PRS receiver, which offers rewinding slots,
- the external party \mathcal{B} , which is a barrier to rewinding.

As usual, we assume (for presentational simplicity) that PRS messages from (and to) \mathcal{A} are of the form $(\mathbf{Type}, \text{values})$. Thus, we have following message types, where m is the “actual” message. Firstly, messages which are irrelevant to the rewinding schedule:

- (\mathbf{Init}, s) : Initiate PRS session s .
- (\mathbf{Other}, s, m) : These are all other messages (to and from \mathcal{A}) which are not covered below (e.g. the commitment phase step 1).

The messages related to the challenge-response phase/the slots, and message to the external party \mathcal{B} , are used in the rewinding schedule. These are the following:

- (\mathbf{Start}, s, m) : Start of challenge-response in PRS session s .⁴ (Sent by \mathcal{A} .)
- (\mathbf{Chall}_i, s, m) : Challenge for i -th slot of PRS session s . (Sent by PRS oracle.)
- (\mathbf{Resp}_i, s, m) : Response to i -th slot of PRS session s . (Sent by \mathcal{A} .)
- (\mathbf{End}, s, m) : Extracted PRS session result. (Sent by PRS oracle.)
- $(\mathbf{ExtSend}_i, m)$: The i -th message from \mathcal{A} to \mathcal{B} . (Sent by \mathcal{A} .)
- $(\mathbf{ExtResp}_i, m)$: The i -th response from \mathcal{B} to \mathcal{A} . (Sent by PRS oracle.)

We assume for simplicity that \mathcal{A} is well-behaved, i.e. never sends unexpected or malformed messages, skips steps, etc. Moreover, we note (and will see) that the presence of a CRS does not affect the rewinding schedule of PRS extractor, which only depends on challenge slots.

⁴ Either m is the last message of the commitment phase step 1. Or one lets (\mathbf{Other}, s, m) finish that step and sets $m = \perp$ here. Either way, the message is a “start marker” initiating the first challenge. The choice does not affect the rewinding schedule.

With notation in place, we recall the rewinding schedule of [Goy+15] (adapted to our notation). We ignore the messages (\mathbf{Init}, s) and (\mathbf{Other}, s, m) in the description, as they are simply handled “honestly” and do not affect the rewinding schedule in any way. Rewinds only happen to sample fresh challenges (\mathbf{Chall}_i, s, m) and gather (fresh) responses, while respecting external messages $(\mathbf{ExtResp}_i, m)$ which cannot be rewound. We use the PRS preamble as defined in Construction 1. The **procedure** $\text{recurse}(t, \text{st}, \mathcal{T}, f, \text{aux}, \text{id})$ is recursively defined with base case for step size $t = 1$. We assume w.l.o.g. that t is a power of 2.

Base case: **procedure** $\text{recurse}(1, \text{st}, \mathcal{T}, f, \text{aux}, \text{id})$

1. If the next message is (\mathbf{Start}, s, m) , start a new session s :
 - Send $(\mathbf{Chall}_1, s, r_1)$ for $r_1 \leftarrow \mathcal{C}$, where $\mathcal{C} = \{0, 1\}^\kappa$.
 - Add $((s, \text{id}), 1, r_1, m)$ to \mathcal{T} .
2. If the next message is (\mathbf{Resp}_i, s, m) :
 - If the simulated PRS receiver in session s would abort (due to a failing check), abort session s and add (s, i, \perp, \perp) to \mathcal{T} . Else continue.
 - If $i \in \{1, \dots, \ell\}$
 - Add (s, i, r_i, m) to \mathcal{T}
 - If $i < \ell$: Send $(\mathbf{Chall}_i, s, r_i)$ for $r \leftarrow \mathcal{C}$.
 - If $i = \ell$: Send $(\mathbf{End}, s, \text{extract}(s, \text{id}, \mathcal{T}, \text{aux}))$.
3. If the next message is $(\mathbf{ExtSend}_i, m)$:
 - If $f = 0$, i.e. this is a *look-ahead* thread, **return** (st, \mathcal{T}) . (Early return.)
 - If $f = 1$, i.e. this is the *main thread*, then:
 - For every live session $s \in \text{LIVE}(\text{st})$ do:
 - * $\times_{s, \text{id}} = 1$,
 - * for every block id' that contains the block id , set $\times_{s, \text{id}'} = 1$.
 - Send m to \mathcal{B} and receive response m' . Forward $(\mathbf{ExtResp}_i, m')$ to \mathcal{A} .
4. If not early returned, update state st to current state of \mathcal{A} and return (st, \mathcal{T}) .

Note that whenever we record a response of sessions s , we also remember the identity id of the block where this has occurred. This is used to disambiguate sessions which occur in parallel in different look-ahead threads. For example, the third session on a look-ahead thread and on the main thread may be completely different sessions. This ensures that (s, id) is a unique identifier across all threads.

Recursive case: **procedure** $\text{recurse}(t, \text{st}, \mathcal{T}, f, \text{aux}, \text{id})$

- // Rewind the first half twice:
1. $(\text{st}_1, \mathcal{T}_1) \leftarrow \text{recurse}(t/2, \text{st}, \mathcal{T}, 0, \text{aux}, \text{id} \circ 1)$ (look-ahead block)
 2. Let $\text{aux}_2 = (\text{aux}, \mathcal{T}_1 \setminus \mathcal{T})$
 $(\text{st}_2, \mathcal{T}_2) \leftarrow \text{recurse}(t/2, \text{st}, \mathcal{T}, f, \text{aux}_2, \text{id} \circ 2)$ (main block)
- // Rewind the second half twice:
3. Let $\mathcal{T}^* = \mathcal{T}_1 \cup \mathcal{T}_2$
 $(\text{st}_3, \mathcal{T}_3) \leftarrow \text{recurse}(t/2, \text{st}_2, \mathcal{T}^*, 0, \text{aux}, \text{id} \circ 3)$ (look-ahead block)
 4. Let $\text{aux}_4 = (\text{aux}, \mathcal{T}_3 \setminus \mathcal{T}^*)$
 $(\text{st}_4, \mathcal{T}_4) \leftarrow \text{recurse}(t/2, \text{st}_2, \mathcal{T}^*, f, \text{aux}_4, \text{id} \circ 4)$ (main block)

Extraction: **procedure** $\text{extract}(s, \text{id}, \mathcal{T}, \text{aux})$

1. Search in \mathcal{T} for a pair of $((s, \text{id}'), i, r_i, m), ((s, \text{id}''), i, r'_i, m')$ with $r'_i \neq r_i$ and such that id', id'' lie after id . If found, extract that pair and return an extracted value.
2. If no such pair exists in \mathcal{T} , consider every block id_1 for which $\times_{s, \text{id}_1} = 1$.
 - Let id'_1 be the sibling⁵ of id_1 with input/output tables $\mathcal{T}_{\text{in}}, \mathcal{T}_{\text{out}}$ respectively.
 - Attempt to extract (as before) from $\text{aux}_{\text{id}'_1} := \mathcal{T}_{\text{in}} \setminus \mathcal{T}_{\text{out}}$.
 - If all attempts fail, return `ExtFail`, otherwise return the extracted value.

Remark 3 (Ambiguous extraction). In $\text{extract}(s, \text{id}, \mathcal{T}, \text{aux})$, it can happen that multiple *distinct* values could be extracted, e.g. because the PRS preamble was inconsistent, and different values were shared in different slots or within a slot. We do not specify which value should be extracted in this case; any choice is fine.

Remark 4 ((Hierarchically) Structured randomness). The procedure `recurse` is deterministic in all recursive calls, except the base calls. It will be helpful to assume that `recurse` interprets its randomness in a structured manner into disjoint/independent parts as follows:

- A tuple $(r_{\text{id}}^{\text{Chall}})_{\text{id}}$ which specifies challenge messages the for slot in atomic block id , i.e. base call ids (i.e. strings in $\{1, 2, 3, 4\}^{\log(t)}$).
- A tuple $(r_{\text{id}}^{\text{Other}})_{\text{id}}$ which specifies randomness for all other probabilistic computations, e.g. the receiver randomness used in base commitments in the PRS commitment phase.

In particular, it is possible to *identify and fix the randomness of the main thread*, and thus all messages and challenges “sent” by `recurse` on the main thread. This separation of randomness into atomic blocks will be conceptually helpful later.

4.2 Composition-Order Invariance of \mathcal{O}_{CCA}

In this section, we prove the k -robust composition-order invariance of \mathcal{O}_{CCA} from Theorem 1. For this, we consider an adversary \mathcal{A} with access to \mathcal{O}_{CCA} and an external protocol \mathcal{B} , so that the interaction $\langle \mathcal{B}, \mathcal{A} \rangle$ has at most k rounds.

Remark 5. The switch from \mathcal{A} interacting with external \mathcal{B} to $\langle \mathcal{B}, \mathcal{A} \rangle$ as a composed system effectively corresponds to making the previously external messages between \mathcal{A} and \mathcal{B} “internal”, hence they are not visible to `recurse` anymore. For example, in a system composed of three machines and a pseudo-oracle \mathcal{O}_{CCA} , we can compose the system in several ways:

- $\langle \mathcal{C}, \langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\text{CCA}}} \rangle \rangle$: Here, all messages from \mathcal{A} and to \mathcal{B} or \mathcal{C} are external (for \mathcal{O}_{CCA}), whereas \mathcal{C} and \mathcal{B} are the single external entity to \mathcal{O}_{CCA} . Indeed, this is equivalent to $\langle \mathcal{C} \parallel \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\text{CCA}}} \rangle$.
- $\langle \mathcal{C}, \langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}_{\text{CCA}}} \rangle$: Here, all messages from \mathcal{A} or \mathcal{B} (i.e., from the composed system $\langle \mathcal{B}, \mathcal{A} \rangle$) to \mathcal{C} are external (for \mathcal{O}_{CCA}).
- $\langle \mathcal{B}, \langle \mathcal{C}, \mathcal{A} \rangle^{\mathcal{O}_{\text{CCA}}} \rangle$: Same as above, with roles of \mathcal{B} and \mathcal{C} swapped.

⁵ The sibling of a block/identity if the other block/identity in the paired calls, e.g. the sibling of $\text{id} \circ 1$ is $\text{id} \circ 2$ and vice versa.

- $\langle \mathcal{B}, \langle \mathcal{C}, \mathcal{A} \rangle \rangle^{\mathcal{O}_{\text{CCA}}}$: Here, there are no external messages. Indeed, this is equivalent to $(\mathcal{C} \parallel \mathcal{B} \parallel \mathcal{A})^{\mathcal{O}_{\text{CCA}}}$.

Before analyzing \mathcal{O}_{CCA} from Theorem 1 further, note that it generates the setup and outputs the receiver’s view $\text{view}_{\mathcal{R}_s}$, unlike the PRS extractor. This does not affect security in any way. Indeed, setup generation is clearly not a problem. And, perhaps surprisingly, outputting the receiver’s view also poses no problem in the security reduction.

Lemma 1. *Let $\mathcal{A}, \mathcal{B}, \mathcal{O}_{\text{CCA}}$ as above and recall that $\langle \mathcal{B}, \mathcal{A} \rangle$ has at most k rounds. Suppose that COM' has stateless receiver. Suppose $M = 2^m$ is an upper bound on the number of messages \mathcal{A} to the PRS oracle or to \mathcal{B} . Let T be an upper bound of the number of sessions started by \mathcal{A} on the main thread. Define the random variables*

- $\text{out}_1(\kappa, x, y, z)$ as $\text{out}_{\mathcal{B}, \mathcal{A}}(\langle \mathcal{B}(x), \mathcal{A}^{\mathcal{O}_{\text{CCA}}}(y) \rangle(1^\kappa, z))$, and
- $\text{out}_2(\kappa, x, y, z)$ as $\text{out}_{\mathcal{B}, \mathcal{A}}(\langle \mathcal{B}(x), \mathcal{A}(y) \rangle^{\mathcal{O}_{\text{CCA}}}(1^\kappa, z))$.

Then there exists an adversary $\mathcal{A}_{\text{COM}'}$ against the binding property of COM' with expected⁶ run-time bounded roughly by the (strict) runtime of extractor $\mathcal{E}^{\langle \mathcal{B}, \mathcal{A} \rangle}$ (cf. Lemma 3). Concretely, if $\langle \mathcal{B}, \mathcal{A} \rangle$ has worst-case run-time S , then $\mathcal{E}^{\langle \mathcal{B}, \mathcal{A} \rangle}$ and $\mathcal{A}_{\text{COM}'}$ has expected run-time bounded roughly by $2 \cdot M^2 S$. In particular, if \mathcal{B} and \mathcal{A} are PPT, then $\mathcal{A}_{\text{COM}'}$ is expected polynomial time. For $\mathcal{A}_{\text{COM}'}$, it holds that for all $\kappa \in \mathbb{N}$ and all $z \in \{0, 1\}^$:*

$$\begin{aligned} \Delta(\text{out}_1(\kappa, x, y, z), \text{out}_2(\kappa, x, y, z)) &\leq 2 \cdot (2^{-\ell + (k+2) \log(M)} + M^2/|\mathcal{C}|) + 2^{-\kappa} \\ &\quad + \frac{1}{T \cdot \text{poly}} \cdot \text{Adv}_{\text{COM}', \mathcal{A}_{\text{COM}'}}^{\text{bind}}(\kappa, z) \end{aligned}$$

where $\text{poly}(\kappa) = \text{poly}_{\text{AoK}}(\kappa) + \kappa \cdot \ell(\kappa)$ and poly_{AoK} is a bound on the number of commitments made during in the AoK step.

The proof idea is straightforward: Whenever `extract` is called for a session which is *visible on the main thread* and thus part of the view, the extracted value must be the same for both $\Pi_1 = \langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\text{CCA}}} \rangle$ and $\Pi_2 = \langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}_{\text{CCA}}}$. Indeed, running the extractor, i.e. `recurse`, with fixed randomness for \mathcal{E} , and \mathcal{A} and \mathcal{B} (and fixed inputs x, y) either the outputs of Π_1 and Π_2 are identical, or at some point, the result of an extraction, i.e. the output of \mathcal{O}_{CCA} , *on the main thread* must have been different. Since extraction succeeds with overwhelming probability (statistically), the only failure case is a break of the binding property of the base commitment or an inconsistent PRS commitment, which is a break of the AoK (which reduces to a binding break). Moreover, despite the different rewinding schedules, \mathcal{O}_{CCA} (i.e. \mathcal{E}) sends the same challenges on the main thread. This is a simple consequence of the “disjoint partition of randomness” we postulated in Remark 4. The claim follows. When embedding the binding game on the main thread, one must simulate the receiver in look-ahead threads (in such a way, that

⁶ Expected run-time stems from extraction of the AoK via rewinding. It can be traded for only one rewind, hence strict PPT, but with a quadratic loss in advantage.

the PRS analysis still applies). This is where the *stateless receiver* property is used, as it ensures that anyone can continue the receiver's interactions. A more detailed proof is provided below.

Proof. Suppose w.l.o.g. that \mathcal{A} resp. \mathcal{B} are deterministic and fix inputs x resp. y . Draw and fix the random tape r for `recurse` and all other randomness of \mathcal{O}_{CCA} . Note that we assume (Remark 4) that the randomness r is of the form $r = (r_{\text{id}})_{\text{id} \in \{1,2,3,4\}^{\log(M)}}$ such that all atomic blocks use disjoint randomness. W.l.o.g., the rewinding sets $t = M$. Let $\Pi_1 = \langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\text{CCA}}} \rangle$ and $\Pi_2 = \langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\text{CCA}}} \rangle$. Since the extractor \mathcal{E} , i.e., `recurse`, is used to implement the PRS extraction in \mathcal{O}_{CCA} , we will talk about *threads* of Π_1 resp. Π_2 by an abuse of notation.

Now, compare the main thread on Π_1 and Π_2 . Since randomness r for `recurse` is fixed and \mathcal{A} and \mathcal{B} assumed deterministic, we observe (by induction) that:

1. If all messages received by \mathcal{A} or \mathcal{B} in Π_1 resp. Π_2 on the main thread are identical, then the next message of \mathcal{A} or \mathcal{B} will again be identical
2. Only \mathcal{O}_{CCA} may send messages which are not identical in Π_1 and Π_2 . We say, that (the responses of \mathcal{O}_{CCA} in) Π_1 and Π_2 *diverge*.

Thus, we will in the following view the execution of Π_1 and Π_2 in parallel and in lock-step on the main thread, until they diverge.

There are two possible cases for diverging responses on the main thread: For some session s , the AoK was accepting but

1. Extraction via `recurse` failed for one of Π_1 or Π_2 (but not both). Denote such an extraction failure event in Π_i by F_i for $i = 1, 2$. Clearly, the event by $F_1 \vee F_2$ is a superset of this case of divergence.
2. Extraction via `recurse` succeeds for both sessions, but extracted values are unequal, i.e. $v_1 \neq v_2$. Denote this event by F_{\neq} .

By Lemma 3 and a union bound, the probability of an extraction failure for the run with Π_1 or Π_2 is at most

$$\Pr[F_1 \vee F_2] \leq 2 \cdot (2^{-\ell + (k+2)\log(M)} + M^2/|\mathcal{C}|).$$

In the following, we consider *modified* outputs $\text{out}_1, \text{out}_2$ which always output 0 if $F_1 \vee F_2$ occurred. (For this, they run both Π_1 and Π_2 with the same randomness.) The change in statistical distance is at most $\Pr[F_1 \vee F_2]$. Thus, from now on, we can ignore the failure case $F_1 \vee F_2$.

Next, we show following claim:

Claim 3. $\Pr[F_{\neq}] \leq 2^{-\kappa} + \frac{1}{\text{poly}\cdot T} \cdot \text{Adv}_{\text{COM}', \mathcal{A}_{\text{COM}'}}^{\text{bind}}(\kappa)$.

The lemma then immediately follows. To prove Claim 3, first denote by s^* the first session (on the main thread) where the divergence of Π_1 and Π_2 occurs. Consider the experiment where after the occurrence of F_{\neq} , the corresponding AoK gets extracted, and if extraction is successful, let $(d''_{i,j})_{i=1}^{\kappa}$ be the extracted decommitments in session s^* slot $j \in \{1, \dots, \ell\}$. Note already here that, even though the extraction of the AoK uses rewinding, it will never rewind *before* the PRS commitment phase step 1 ended for session s^* (on the main thread).

Observe that F_{\neq} implies that at least one of the following is true.

- If the AoK extraction fails, then for our concrete instantiation, either
 - no two different responses were found (during rewinding), which happens with negligible probability $2^{-\kappa}$.⁷ Call this event F_{coll} .
 - or two different responses were found but they were inconsistent, i.e. the decommitments they had in common were not all to the same values. Thus, this yields a binding break.
- If AoK extraction succeeds, then at least one extracted decommitment $d''_{i,j}$ unveils to a different value than recurse extracted for Π_1 or Π_2 . Again, this yields a binding break.

Now, we construct an adversary $\mathcal{A}_{\text{COM}'}$ against the binding property of COM' . Note that $\mathcal{A}_{\text{COM}'}$ has to rewind \mathcal{A} , but cannot rewind the embedded binding game. As a consequence, arguing that $\mathcal{A}_{\text{COM}'}$ correctly simulates the experiment requires some care. Unsurprisingly, $\mathcal{A}_{\text{COM}'}$ embeds its binding challenge in a random instance of COM' on the main thread. It does so by passing the (external) messages for COM' from \mathcal{A} to the game and returning the challenge receiver's responses. However, $\mathcal{A}_{\text{COM}'}$ must also play the receiver in look-ahead threads, where it cannot embed the binding game anymore. That is, $\mathcal{A}_{\text{COM}'}$ must procure responses for \mathcal{A} whose distribution is identical to that of the receiver of COM' (with the same state as that in the “past” of the thread under consideration). For this, we exploit that COM' has *stateless receiver*: Thus, $\mathcal{A}_{\text{COM}'}$ can simply continue the execution of any COM' receiver. Observe, that since the randomness in all atomic blocks is independent (cf. Remark 4), the embedding of the COM' challenger in the main thread and computing the stateless receiver responses in look-ahead threads does not affect the distribution (in fact, it is possible to map random tapes from one execution to the other and vice versa). Thus, the probability for F_{\neq} is unchanged. In full, $\mathcal{A}_{\text{COM}'}$ works as follows:

- Pick a random commitment index t^* on the main thread. That is, pick a random session $s^* \leftarrow \{1, \dots, T\}$ and index $i^* \xleftarrow{\$} \{1, \dots, \text{poly}(\kappa)\}$, where $\text{poly}(\kappa) = \text{poly}_{\text{AoK}}(\kappa) + \kappa \cdot \ell(\kappa)$ is an upper bound for the number of COM' commitments made in a PRS commitment phase (i.e. both in step 1 and the AoK).
- Run Π_1 and Π_2 in parallel and synchronized on the main thread.
- If F_{\neq} occurs before session s^* , output \perp to the challenger.⁸
- Emulate the rest of the extractor/rewinding schedule essentially unchanged.
- Embed the binding challenge in session s^* and commitment with index i^* .
- After session s^* , extract the AoK (via rewinding) and output a potential binding break for commitment i^* to the challenger, or \perp if none occurred or extraction of the AoK failed. Observe that:

⁷ Resampling challenges uniformly with replacement, the probability of a collision is $1/n$ if there are n accepting challenges, and $n/2^\kappa$ is the probability that the verifier accepted the AoK (and extraction was started). Thus $\max_{n=1}^{2^\kappa} 2^{-\kappa} n/n = 2^{-\kappa}$ is an upper bound on failure.

⁸ If $F_1 \vee F_2$ occurs before F_{\neq} , the modified experiment immediately outputs 0, so F_{\neq} will not occur and this case is irrelevant.

- The rewinding-based AoK extraction occurs strictly *after* the embedded challenge commitment completed, hence $\mathcal{A}_{\text{COM}'}$ never attempts to rewinding the challenger.
- The reduction $\mathcal{A}_{\text{COM}'}$ never provides $(\text{End}, s^*, v_{s^*}, \text{view}_{\mathcal{R}_{s^*}})$ to \mathcal{A} . Indeed, it could not provide $\text{view}_{\mathcal{R}_{s^*}}$, since in general, $\text{view}_{\mathcal{R}_{s^*}}$ is only known to the binding challenger.

Overall, this yields our adversary $\mathcal{A}_{\text{COM}'}$ against the multi-binding game with the claimed advantage. In more detail: Let B be the event that a binding break is found on the main thread when the AoK for the first diverging session s^* is extracted. We find that

$$\Pr[F_{\neq}] \leq \Pr[F_{\text{coll}}] + \Pr[B] \leq 2^{-\kappa} + \frac{1}{\text{poly} \cdot T} \cdot \text{Adv}_{\text{COM}', \mathcal{A}_{\text{COM}'}}^{\text{bind}}(\kappa).$$

Putting everything together, we find that for all $\kappa \in \mathbb{N}$ and all $z \in \{0, 1\}^*$, it holds that

$$\begin{aligned} \Delta(\text{out}_1(\kappa, x, y, z), \text{out}_2(\kappa, x, y, z)) &\leq \Pr[F_1 \vee F_2 \vee F_{\text{coll}} \vee B] \\ &\leq 2 \cdot (2^{-\ell + (k+2)\log(M)} + M^2/|\mathcal{C}|) + 2^{-\kappa} \\ &\quad + \frac{1}{\text{poly} \cdot T} \cdot \text{Adv}_{\text{COM}', \mathcal{A}_{\text{COM}'}}^{\text{bind}}(\kappa, z) \end{aligned}$$

This proves the claimed advantage of $\mathcal{A}_{\text{COM}'}$.

Lastly, observe that if \mathcal{B} and \mathcal{A} are PPT, then Π_1 and Π_2 are (overall) PPT algorithms, since \mathcal{O}_{CCA} is k -robust pseudo-PPT. In particular, rewinding them for AoK extraction is efficient (and the expected time bound is roughly double the worst-case time since in expectation 1 rewind happens). \square

5 Framework and Notion

With UC security and its variants, all entities keep their run-time complexity throughout the whole execution, making them unsuitable to analyze the security of protocols in a setting where cryptographic hardness assumptions may lose their validity.

In order to circumvent the impossibility results of Long-Term Security [MU10], we want to modify the protocol execution such that rewinding-based extraction of long-term hiding commitment schemes is possible in a way that preserves universal composability.

To this end, we take a route similar to [PS04; CLP10] and provide environment and adversary with an entity called the *helper* \mathcal{H} . This *efficient* helper is parameterized with an extractable commitment scheme COM and allows the rewinding-based extraction of instances of COM where the commitment is performed with \mathcal{H} as receiver.

Formally, our notion is cast in the GUC framework [Can+07], allowing both the use of the helper \mathcal{H} as well as other global ideal functionalities. We assume that the reader is familiar with the basic concepts of (G)UC security. For a short overview, see Appendix A. Due to space constraints, this chapter is a short version only. For the full treatment of framework and notion, see Appendix D.

Extracting (Statistically Hiding) Commitments. \mathcal{H} provides an oracle that allows the extraction of commitments (cf. Section 3.2), similar to a CCA oracle. This part is analogous to the helper of [CLP10], with the following differences: The helper of [CLP10] is able to extract statistically binding commitments by inefficient computations. In contrast, we want to consider commitments that are statistically hiding. Such commitments cannot be extracted by brute force, but require different techniques such as an appropriate setup allowing for straight-line extraction (see [MU10] for an example) or rewinding. More specifically, we adapt the rewinding-based extraction techniques of [Goy+15] to our setting, via pseudo-oracles and a suitably adapted analysis in Sections 3 and 4. We provide the helper with the views of all ITIs that may be affected by a performed rewinding.

While we do not (intend to) achieve composability in the plain model, the resulting security notion has properties and limitations similar to Angel-based security [PS04] or UC with super-polynomial helpers [CLP10], e.g. with respect to protocol reusability.

As we will later use commitment schemes in the \mathcal{F}_{CRS} -hybrid model, we have adapted the helper accordingly. When a corrupted party starts a new commitment session with \mathcal{H} , the committed-value oracle \mathcal{O}_{CCA} within \mathcal{H} honestly executes the CRS generation algorithm of the desired commitment scheme and \mathcal{H} returns the resulting CRS ck to the party initiating the session.

As the commitment key is generated honestly, it is guaranteed to be independent from all other commitment keys. Thus, a corrupted party cannot take a key ck' from another session (e.g. where the sender is honest) and have \mathcal{H} extract commitments relative to ck . A similar policy is enforced in [CLP10] by the use of tags, which we omit as they are not necessary (with different sessions distinguished by their commitment key).

To establish meaningful properties, we require the commitment scheme to feature a committed-value oracle which is *black-box* (Definition 4) and *k-robust composition-order invariant and pseudo-PPT* (Definitions 5 and 6). This allows to a) import protocols with appropriate round complexity into our framework without loss of security due to the committed-value oracle and b) prove the security of protocols within our framework by reducing to security properties with a certain (bounded) round complexity. The robustness property guarantees that we can efficiently simulate the committed-value oracle without having to rewind the challenger in a reduction.

The helper \mathcal{H} is formally defined in Definition 9.

Definition 9 (The helper \mathcal{H}). \mathcal{H} is parameterized with 1. a security parameter $\kappa \in \mathbb{N}$, 2. auxiliary input z and 3. a commitment scheme COM with committed-value (pseudo-)oracle \mathcal{O}_{CCA} .

- Upon receiving an input $(\text{corrupt}, P_i, \text{sid})$ from the environment, record $(\text{corrupt}, P_i, \text{sid})$.
- Upon receiving an input $(\text{ext-init}, P_i, \text{sid}, k)$ from a corrupted party P_i in the protocol with SID sid : If there is a recorded session (P_i, sid, k) , ignore this message. Otherwise, initialize the k -th sub-session of (P_i, sid) with \mathcal{O}_{CCA} and

- receive a setup ck . Record session (P_i, sid, k) and return $(\text{setup}, \text{sid}, k, \text{ck})$ to P_i .
- Upon receiving a message $(\text{ext-mesg}, P_i, \text{sid}, k, m)$ from a corrupted party P_i in the protocol with SID sid : If there is no recorded session (P_i, sid, k) , ignore the message. Otherwise, give input (sid, k, m) to \mathcal{O}_{CCA} , possibly obtain a reply m' . If m' is a special message $(\text{End}, s, w_s, \text{view}_{R_s})$, return $(\text{ext-val}, P_i, \text{sid}, k, w_s)$ to P_i . Otherwise, return $(\text{ext-mesg}, P_i, \text{sid}, k, m')$ to P_i .

In order to allow the helper to perform the rewinding-based extraction, it needs to be provided with the views of the entities to be rewinded. This requires slight changes to the framework, which can be found in Appendix D.1. Knowledge of these changes is not necessary to understand the following. We call the framework resulting from this modification the *UC Security with Rewinding framework*.

5.1 Protocol Emulation

Before stating long-term protocol emulation, we provide the standard notion of computational protocol emulation adapted to our setting.

Definition 10 (UC Security with Rewinding Protocol Emulation). *Let π and ϕ be PPT protocols and let \mathcal{H} be the helper of Definition 9. We say that π Rewinding-UC-emulates ϕ if for all PPT adversaries \mathcal{A} , there exists a PPT simulator \mathcal{S} such that for all \mathcal{H} -aided⁹ balanced PPT environments \mathcal{Z} , there exists a negligible function negl such that for all $\kappa \in \mathbb{N}, z \in \{0, 1\}^*$ it holds that*

$$|\Pr[\text{Exec}(\pi, \mathcal{A}, \mathcal{Z})(\kappa, z) = 1] - \Pr[\text{Exec}(\phi, \mathcal{S}, \mathcal{Z})(\kappa, z) = 1]| \leq \text{negl}(\kappa)$$

If π Rewinding-UC-emulates ϕ , we write $\pi \geq_{\text{R}} \phi$.

We adapt the notion of long-term protocol emulation in our framework in analogy to the established definition due to Müller-Quade and Unruh [MU10]. In contrast to standard UC emulation, long-term emulation allows the environment to output an arbitrary string of polynomial length and requires statistical indistinguishability of the resulting ensembles. Intuitively, this means that all (polynomial-time) hardness assumptions lose their validity after the protocol execution has finished.

To this end, let ExecS denote the random variable that is identically defined to Exec , except that the environment outputs an arbitrary string (of polynomial length).

Definition 11 (Long-term UC Protocol Emulation). *Let π and ϕ be PPT protocols and let \mathcal{H} be the helper of Definition 9. We say that π long-term-Rewinding-UC-emulates ϕ if for all PPT adversaries \mathcal{A} , there exists a PPT*

⁹ We restate the definition of \mathcal{H} -aided environments due to Canetti, Lin, and Pass [CLP16]: a) \mathcal{Z} invokes a single instance of \mathcal{H} immediately after invoking the adversary. b) As soon as a party (i.e. an ITI) P is corrupted (i.e. P receives a **corrupted** message), \mathcal{Z} lets \mathcal{H} know of this fact. \mathcal{H} interacts only with the environment, the adversary, and the corrupted parties.

simulator \mathcal{S} such that for all \mathcal{H} -aided balanced PPT environments \mathcal{Z} , the ensembles $\{\text{ExecS}(\pi, \mathcal{A}, \mathcal{Z})(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^*}$ and $\{\text{ExecS}(\phi, \mathcal{S}, \mathcal{Z})(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^*}$ are statistically indistinguishable.

If π long-term-Rewinding-UC-emulates ϕ , we write $\pi \geq_{\text{R}}^{\text{lt}} \phi$. If π long-term-Rewinding-UC-emulates the ideal protocol of a functionality \mathcal{F} , then we say that π long-term-Rewinding-UC-realizes \mathcal{F} .

Remark 6. In contrast to the definition of long-term security in [MU10], the environment of Definition 11 has access to the helper \mathcal{H} , which provides a committed-value oracle that does not exist in the original definition.

5.2 Properties

We now discuss the properties of our notion, which are mostly similar to the properties of long-term security and UC security with super-polynomial helpers. In particular, the dummy adversary is complete and both notions of protocol emulation are transitive. Like UC security with superpolynomial helpers [CLP10] and long-term security [MU10]), our notion is closed under general concurrent (i.e. universal) composition.

Theorem 2 (Composition Theorem). *Let ρ, π, ϕ be PPT protocols where π and ϕ subroutine-respecting. If π (long-term-) Rewinding-UC-emulates ϕ , then, $\rho^{\phi \rightarrow \pi}$ (long-term-) Rewinding-UC-emulates ρ .*

The proof is very similar to the ones presented in [CLP10; Can+07; MU10] and we omit it.

UC Compatibility. When introducing a new security notion that features modular design, a natural question to ask is which existing protocols (that are secure according some other notion) can be reused.

Let π and ϕ be PPT protocols such that $\pi \geq_{\text{UC}} \phi$. Due to the helper \mathcal{H} , just as in [PS04; CLP10], we cannot hope that we can import an arbitrary UC protocol π securely, i.e. that $\pi \geq_{\text{UC}} \phi$ implies that $\pi \geq_{\text{R}} \phi$. This is because a Rewinding UC environment is more powerful than a normal UC environment due to the access to \mathcal{H} : The committed-value oracle of \mathcal{H} could invalidate assumptions made in the security proof.

Nevertheless, we can show the compatibility with UC security for large classes of protocols, namely those that have less than or equal to k rounds if the committed-value (pseudo-)oracle provided by \mathcal{H} is black-box (Definition 4) k -robust composition-order invariant (Definition 5) and pseudo-PPT (Definition 6). This criterion essentially is the same as in [CLP10], except for the additional requirements for the (pseudo-)oracle.

Theorem 3 (UC Compatibility). *Let \mathcal{H} be the helper that is parameterized with a commitment scheme COM that features an $O(k)$ -robust black-box composition-order invariant pseudo-PPT committed-value (pseudo-)oracle \mathcal{O}_{CCA} ,*

where $k \in O(\text{poly}(\kappa))$. Let ϕ be a subroutine-respecting PPT protocol and let π be a subroutine-respecting PPT protocol with less than or equal to k rounds such that

- $\pi \geq_{\text{stat-UC}} \phi$. Then, $\pi \geq_{\text{R}}^{\text{lt}} \phi$.
- $\pi \geq_{\text{ltUC}} \phi$. Then, $\pi \geq_{\text{R}}^{\text{lt}} \phi$.
- $\pi \geq_{\text{UC}} \phi$. Then, $\pi \geq_{\text{R}} \phi$.

Here, $\geq_{\text{stat-UC}}$ denotes statistical UC emulation, \geq_{ltUC} denotes long-term emulation and \geq_{UC} denotes standard UC emulation.

For the proof, see Appendix D.3.

Of course, compatibility is not limited to the cases mentioned in Theorem 3 and its variants. However, manual proofs may be necessary.

Meaningfulness. Just like the angel in [PS04] or the helper in [CLP10], our helper may negatively affect the security guarantees provided by ideal functionalities. To illustrate this, consider a variant $\mathcal{F}'_{\text{COM}}$ of the ideal functionality for commitments \mathcal{F}_{COM} , which we extend to accept a CRS from the adversary. When the honest committer provides its input v , $\mathcal{F}'_{\text{COM}}$ first checks if the CRS is a valid CRS for the statistically hiding commitment scheme COM of \mathcal{H} ¹⁰. Then, it performs the commit phase with the adversary, acting as an honest committer with input v .

In the presence of \mathcal{H} , $\mathcal{F}'_{\text{COM}}$ provides no meaningful security. The adversary simply can start a new session with the committed-value oracle provided by \mathcal{H} , receiving a valid CRS which it provides to $\mathcal{F}'_{\text{COM}}$. Then, it can forward all commitment-related messages between \mathcal{H} and $\mathcal{F}'_{\text{COM}}$. In the end, the adversary will learn v , i.e. the value committed to by the honest committer, from \mathcal{H} . (The argument for [CLP10; PS04] is analogous.)

Thus, (long-term) Rewinding UC security only guarantees meaningful security for ideal functionalities with less than or equal to k rounds if \mathcal{O}_{CCA} (in \mathcal{H}) is $O(k)$ -robust, pseudo-PPT $O(k)$ -robust composition-order invariant. Note that very similar limitations with respect to the meaningfulness apply to e.g. [CLP10; PS04].

Justification. We now discuss under which circumstances our notion implies existing security notions for (composable) multi-party computation. This is helpful to grasp the (intuitive) security guarantees of (long-term) Rewinding UC security. First, we show that Rewinding UC security implies UC security for a large class of protocols.

Proposition 2 (Justification: UC Security). *Let π, ϕ be PPT protocols such that $\pi \geq_{\text{R}} \phi$ (resp. $\pi \geq_{\text{R}}^{\text{lt}} \phi$) and the simulator never needs to interact with \mathcal{H} on the committed-value oracle for the challenge session. Then, $\pi \geq_{\text{UC}} \phi$ (resp. $\pi \geq_{\text{ltUC}} \phi$).*

¹⁰ Here, we assume that a CRS that leads to a statistically hiding commitment scheme is efficiently recognizable.

For the proof, see Appendix D.3. For the case of ideal functionalities that can be expressed by stand-alone real-ideal security (see e.g. [Gol04]), the following holds regardless of the simulator using the committed-value oracle of \mathcal{H} .

Proposition 3 (Justification: Stand-Alone Security for SFE). *Let \mathcal{H} be a helper with a committed-value oracle that is black-box and $O(1)$ -robust composition-order invariant and pseudo-PPT. Let π be a N -party PPT protocol in the \mathcal{F}_{CRS} -hybrid model such that π (long-term-) Rewinding-UC-realizes \mathcal{F}_{SFE} (with \mathcal{H}) for some function $f : (\{0, 1\}^\kappa)^N \times \{0, 1\}^{\text{poly}(\kappa)} \rightarrow (\{0, 1\}^\kappa)^N$. Then, π securely computes f with abort in the presence of static malicious adversaries.*

In particular, Proposition 3 captures the stand-alone real-ideal security of e.g. zero-knowledge proof systems. The restriction to protocols in the \mathcal{F}_{CRS} -hybrid model can be relaxed to other hybrid functionalities that can be expressed by stand-alone real-ideal security.

We omit the proof of Proposition 3, but note that that the distinguisher in the real-ideal security notion is not provided with a committed-value oracle (corresponding to an Rewinding UC environment that never queries the committed-value oracle of \mathcal{H}). Thus, the (PPT) simulator may only need to extract commitments for its own simulation, which it can do efficiently via rewinding, regardless of the number of rounds of π .

Environmental Friendliness. Similar to [CLP10], our notion partially fulfills the notion of environmental friendliness [CLP13]. Suppose that the committed-value oracle of \mathcal{H} is $O(k)$ -robust, pseudo-PPT $O(k)$ -robust composition-order invariant and that a PPT protocol π (long-term-) Rewinding-UC-realizes an ideal functionality \mathcal{G} . Then, we can show that for every k -round game-based property of a protocol that is executed concurrently (outside the Rewinding UC execution), the protocol π does not affect this game-based property if it is not already affected by \mathcal{G} (in an execution *without* \mathcal{H}). For details, see Appendix D.3.

Impossibility Results. While the addition of the helper \mathcal{H} , which allows the extraction of statistically hiding commitments, suffices to “circumvent” the impossibility results of Müller-Quade and Unruh [MU10], our setting still faces an important impossibility result for *long-term* Rewinding UC.

Theorem 4. *Let \mathcal{F} be a functionality that is long-term revealing (Definition 21) for any party. Then, there is no bilateral¹¹ nontrivial PPT protocol π_{OT} that long-term-Rewinding-UC-realizes \mathcal{F}_{OT} in the \mathcal{F} -hybrid model (assuming ideally authenticated communication).*

Theorem 4 is a direct consequence of the folklore impossibility result of *correct* statistically secure oblivious transfer in the plain model (even with passive security only). For the proof, see Appendix D.3.

¹¹ We recall the definition of a bilateral protocol due to Canetti and Fischlin [CF01]: “[A] protocol π between n parties P_1, \dots, P_n is *bilateral* if all except two parties stay idle and do not transmit messages.”

6 Long-Term-Secure Composable Commitment Scheme

In this chapter, we present the construction of our long-term-secure commitment scheme π_{COM} .

In Section 3, we have constructed a commitment scheme CCACOM that is both CCA-binding (Definition 7) and trapdoor (Definition 8). The equivocation is performed by embedding a (statistically hidden) trapdoor into the CRS. In contrast, the extraction is performed using rewinding. For the committed-value oracle \mathcal{O}_{CCA} , we have been able to establish several important properties such as the *black-box* property (Definition 4) and *k-robust composition-order invariant and pseudo-PPT* (Definitions 5 and 6). In Section 5, we have embedded the committed-value (pseudo-)oracle \mathcal{O}_{CCA} into the UC execution through the helper \mathcal{H} .

Thus, the construction is straight-forward: π_{COM} merely wraps an instance of CCACOM, inheriting its properties. In particular, the equivocation by the simulator is done by choosing an appropriate CRS, while the extraction is performed via the helper.

Even though the construction is simple, the proof needs to carefully deal with the pseudo-oracle \mathcal{O}_{CCA} , which is part of the helper. In order for the reductions to properties of CCACOM to go through, we will make heavy use of the aforementioned properties of \mathcal{O}_{CCA} .

Construction 2 (Protocol π_{COM}) *Parameterized by a security parameter κ and a commitment scheme COM with non-interactive unveil phase.*

- On input $(\text{commit}, \text{sid}, v)$ for C:
 1. C and R execute COM with SID $(\text{sid}||\text{COM})$ and input v for the committer. Let d denote the unveil information. If the sub-party of the receiver in COM accepts, R outputs $(\text{committed}, \text{sid})$. Subsequent commit inputs are ignored.
- On input $(\text{unveil}, \text{sid})$ for C:
 1. C sends $(\text{unveil}, \text{sid}, v, d)$ to R.
 2. R outputs $(\text{unveil}, \text{sid}, v)$ if the commitment opens to v using unveil information d . Otherwise, it halts without output.

We can now state our main theorem. In the following, we always assume that the protocol under consideration, helper and ideal functionality are consistent, i.e. parameterized with the same commitment scheme COM. The proof of following theorem is in Appendix E.

Theorem 5. *Let \mathcal{O}_{CCA} be a black-box committed-value pseudo-oracle for the commitment scheme COM. If COM is a CCA-binding and trapdoor commitment scheme w.r.t. \mathcal{O}_{CCA} and has an appropriate message space, then π_{COM} long-term-Rewinding-UC-realizes \mathcal{F}_{COM} .*

7 Applications

We present several applications of our composable long-term-secure commitment scheme. For a full treatment, see Appendix F.

7.1 Zero-Knowledge and Commit-and-Prove

By plugging our long-term-secure commitment scheme into an appropriate zero-knowledge proof system with statistical UC security in the \mathcal{F}_{COM} -hybrid model (e.g. [CF01]), we can long-term-Rewinding-UC-realize \mathcal{F}_{ZK} . The resulting protocol thus features composable statistical zero-knowledge and knowledge soundness against computationally bounded provers.

Using a similar approach, we obtain a protocol that long-term-Rewinding-UC-realizes the ideal functionality \mathcal{F}_{CP} for commit-and-prove (Definition 15) for a bounded number of proofs per instance.

7.2 Two-Party Computation with Long-term Security for One Party

Even given a commitment scheme that is long-term-Rewinding-UC-secure, we cannot hope to achieve long-term secure oblivious transfer from long-term-revealing setups (cf. Theorem 4), which also rules out general secure two-party computation with long-term security for both parties.

By combining \mathcal{F}_{CP} with an appropriate oblivious transfer (OT) protocol that provides statistical security for *one* party, e.g. the dual-mode construction of [PVW08], we obtain a protocol for composable OT where one party is protected with long-term security.

Using a very similar approach, we can construct protocols for composable general two-party computation where one party enjoys long-term security.

Unfortunately, meaningfully defining security in such a setting is not straightforward: The simulation of an (honest) party that is only given computational security must depend on this party's secrets, which are (not even indirectly) available for the simulator.

For the constructions and further discussions, see Appendix F.2.

8 Conclusion

Previous constructions for protocols with composable long-term security required hardware-based setups, often making them impractical due to the necessary distribution of the setup. In particular, natural setups such as common reference strings were shown to be insufficient to achieve long-term security.

We circumvent this impossibility result by enabling a rewinding-based extraction through the introduction of pseudo-oracles. Towards this, we faced and solved many technical hurdles to construct a robust, extractable, and well-behaved CCA-commitment scheme. With rewinding-based extraction at hand, we are the first to construct a statistically hiding and composable long-term-secure commitment scheme from standard polynomial-time hardness assumptions solely in the \mathcal{F}_{CRS} -hybrid model, i.e. without the use of hardware assumptions.

We give several applications of our commitment scheme, including composable oblivious transfer with long-term security for *one* party in the \mathcal{F}_{CRS} -hybrid model and showed how this approach can be extended to general two-party computation with similar guarantees. Due to impossibility results within our security notion, this is the best security one can hope for, unless stronger setups are used.

Acknowledgements. Astrid Ottenhues: This work was supported by funding from the German Federal Ministry of Education and Research (BMBF) under the projects “PQC4MED” (ID 16KIS1044) and “Sec4IoMT” (ID 16KIS1692). Michael Kloock, Jeremias Mechler, Jörn Müller-Quade, Markus Raiber: This work was supported by funding from the topic Engineering Secure Systems of the Helmholtz Association (HGF) and by KASTEL Security Research Labs.

References

- [AL17] Gilad Asharov and Yehuda Lindell. “A Full Proof of the BGW Protocol for Perfectly Secure Multiparty Computation”. In: *Journal of Cryptology* 30.1 (Jan. 2017), pp. 58–151. DOI: 10.1007/s00145-015-9214-4.
- [Bea97] Donald Beaver. “Commodity-Based Cryptography (Extended Abstract)”. In: *29th Annual ACM Symposium on Theory of Computing*. El Paso, TX, USA: ACM Press, 1997, pp. 446–455. DOI: 10.1145/258533.258637.
- [BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. “Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)”. In: *20th Annual ACM Symposium on Theory of Computing*. Chicago, IL, USA: ACM Press, 1988, pp. 1–10. DOI: 10.1145/62212.62213.
- [Can01] Ran Canetti. “Universally Composable Security: A New Paradigm for Cryptographic Protocols”. In: *42nd Annual Symposium on Foundations of Computer Science*. Las Vegas, NV, USA: IEEE Computer Society Press, 2001, pp. 136–145. DOI: 10.1109/SFCS.2001.959888.
- [Can+02] 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*. Montréal, Québec, Canada: ACM Press, 2002, pp. 494–503. DOI: 10.1145/509907.509980.
- [Can+07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. “Universally Composable Security with Global Setup”. In: *TCC 2007: 4th Theory of Cryptography Conference*. Ed. by Salil P. Vadhan. Vol. 4392. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, 2007, pp. 61–85. DOI: 10.1007/978-3-540-70936-7_4.
- [CF01] Ran Canetti and Marc Fischlin. “Universally Composable Commitments”. In: *Advances in Cryptology – CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2001, pp. 19–40. DOI: 10.1007/3-540-44647-8_2.
- [CLP10] Ran Canetti, Huijia Lin, and Rafael Pass. “Adaptive Hardness and Composable Security in the Plain Model from Standard Assumptions”. In: *51st Annual Symposium on Foundations of Computer Science*.

- Las Vegas, NV, USA: IEEE Computer Society Press, 2010, pp. 541–550. DOI: 10.1109/FOCS.2010.86.
- [CLP13] Ran Canetti, Huijia Lin, and Rafael Pass. “From Unprovability to Environmentally Friendly Protocols”. In: *54th Annual Symposium on Foundations of Computer Science*. Berkeley, CA, USA: IEEE Computer Society Press, 2013, pp. 70–79. DOI: 10.1109/FOCS.2013.16.
- [CLP16] Ran Canetti, Huijia Lin, and Rafael Pass. “Adaptive Hardness and Composable Security in the Plain Model from Standard Assumptions”. In: *SIAM J. Comput.* 45.5 (2016), pp. 1793–1834. DOI: 10.1137/110847196. URL: <https://doi.org/10.1137/110847196>.
- [DKM11] Nico Döttling, Daniel Kraschewski, and Jörn Müller-Quade. “Unconditional and Composable Security Using a Single Stateful Tamper-Proof Hardware Token”. In: *TCC 2011: 8th Theory of Cryptography Conference*. Ed. by Yuval Ishai. Vol. 6597. Lecture Notes in Computer Science. Providence, RI, USA: Springer, Heidelberg, Germany, 2011, pp. 164–181. DOI: 10.1007/978-3-642-19571-6_11.
- [DN02] Ivan Damgård and Jesper Buus Nielsen. “Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor”. In: *Advances in Cryptology – CRYPTO 2002*. Ed. by Moti Yung. Vol. 2442. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2002, pp. 581–596. DOI: 10.1007/3-540-45708-9_37.
- [GK90] Oded Goldreich and Hugo Krawczyk. “On the Composition of Zero-Knowledge Proof Systems”. In: *ICALP90*. Ed. by Mike Paterson. Vol. 443. Lecture Notes in Computer Science. Springer, 1990, pp. 268–282. DOI: 10.1007/BFb0032038. URL: <https://doi.org/10.1007/BFb0032038>.
- [Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*. Vol. 2. Cambridge, UK: Cambridge University Press, 2004. ISBN: ISBN 0-521-83084-2 (hardback).
- [Goy+15] Vipul Goyal, Huijia Lin, Omkant Pandey, Rafael Pass, and Amit Sahai. “Round-Efficient Concurrently Composable Secure Computation via a Robust Extraction Lemma”. In: *TCC 2015: 12th Theory of Cryptography Conference, Part I*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9014. Lecture Notes in Computer Science. Warsaw, Poland: Springer, Heidelberg, Germany, 2015, pp. 260–289. DOI: 10.1007/978-3-662-46494-6_12.
- [GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. “Leveled Fully Homomorphic Signatures from Standard Lattices”. In: *47th Annual ACM Symposium on Theory of Computing*. Ed. by Rocco A. Servedio and Ronitt Rubinfeld. Portland, OR, USA: ACM Press, 2015, pp. 469–477. DOI: 10.1145/2746539.2746576.
- [HW09] Susan Hohenberger and Brent Waters. “Short and Stateless Signatures from the RSA Assumption”. In: *Advances in Cryptology –*

- CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2009, pp. 654–670. DOI: 10.1007/978-3-642-03356-8_38.
- [Kiy20] Susumu Kiyoshima. “Statistical Concurrent Non-Malleable Zero-Knowledge from One-Way Functions”. In: *Journal of Cryptology* 33.3 (July 2020), pp. 1318–1361. DOI: 10.1007/s00145-020-09348-x.
- [Mag+22] Bernardo Magri, Giulio Malavolta, Dominique Schröder, and Dominique Unruh. “Everlasting UC Commitments from Fully Malicious PUFs”. In: *J. Cryptol.* 35.3 (2022), p. 20.
- [MU10] Jörn Müller-Quade and Dominique Unruh. “Long-Term Security and Universal Composability”. In: *Journal of Cryptology* 23.4 (Oct. 2010), pp. 594–671. DOI: 10.1007/s00145-010-9068-8.
- [Nie03] Jesper Buus Nielsen. “On Protocol Security in the Cryptographic Model”. English. PhD thesis. 2003.
- [Orl+14] Claudio Orlandi, Rafail Ostrovsky, Vanishree Rao, Amit Sahai, and Ivan Visconti. “Statistical Concurrent Non-malleable Zero Knowledge”. In: *TCC 2014: 11th Theory of Cryptography Conference*. Ed. by Yehuda Lindell. Vol. 8349. Lecture Notes in Computer Science. San Diego, CA, USA: Springer, Heidelberg, Germany, 2014, pp. 167–191. DOI: 10.1007/978-3-642-54242-8_8.
- [Pas03] Rafael Pass. “Simulation in Quasi-Polynomial Time, and Its Application to Protocol Composition”. In: *Advances in Cryptology – EUROCRYPT 2003*. Ed. by Eli Biham. Vol. 2656. Lecture Notes in Computer Science. Warsaw, Poland: Springer, Heidelberg, Germany, 2003, pp. 160–176. DOI: 10.1007/3-540-39200-9_10.
- [Ped92] Torben P. Pedersen. “Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing”. In: *Advances in Cryptology – CRYPTO’91*. Ed. by Joan Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 1992, pp. 129–140. DOI: 10.1007/3-540-46766-1_9.
- [PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. “Concurrent Zero Knowledge with Logarithmic Round-Complexity”. In: *43rd Annual Symposium on Foundations of Computer Science*. Vancouver, BC, Canada: IEEE Computer Society Press, 2002, pp. 366–375. DOI: 10.1109/SFCS.2002.1181961.
- [PS04] Manoj Prabhakaran and Amit Sahai. “New notions of security: Achieving universal composability without trusted setup”. In: *36th Annual ACM Symposium on Theory of Computing*. Ed. by László Babai. Chicago, IL, USA: ACM Press, 2004, pp. 242–251. DOI: 10.1145/1007352.1007394.
- [PTV14] Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkatasubramanian. “Concurrent Zero Knowledge, Revisited”. In: *Journal of Cryptology* 27.1 (Jan. 2014), pp. 45–66. DOI: 10.1007/s00145-012-9137-2.

- [PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. “A Framework for Efficient and Composable Oblivious Transfer”. In: *Advances in Cryptology – CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2008, pp. 554–571. DOI: 10.1007/978-3-540-85174-5_31.

Supplementary Material

Appendix

Table of Contents

Composable Long-Term Security with Rewinding	1
<i>Robin Berger, Brandon Broadnax, Michael Kloof, Jeremias Mechler, Jörn Müller-Quade, Astrid Ottenhues, Markus Raiber</i>	
1 Introduction	2
1.1 Outline and Contribution	4
1.2 Technical Overview	4
1.3 Related Work	7
2 Preliminaries	9
2.1 Commitment Schemes	9
3 A Statistically Hiding Concurrently Extractable and Equivocal Commitment Scheme	10
3.1 Pseudo-Oracles	10
3.2 Properties of Commitment Schemes	12
3.3 Constructions	14
4 Analysis of the Committed-Value Oracle \mathcal{O}_{CCA}	17
4.1 The Rewinding Schedule from [Goy+15]	17
4.2 Composition-Order Invariance of \mathcal{O}_{CCA}	19
5 Framework and Notion	23
5.1 Protocol Emulation	25
5.2 Properties	26
6 Long-Term-Secure Composable Commitment Scheme	29
7 Applications	29
7.1 Zero-Knowledge and Commit-and-Prove	30
7.2 Two-Party Computation with Long-term Security for One Party	30
8 Conclusion	30
A A Short Introduction to (G)UC	37
A.1 Important Ideal Functionalities	39
B Preliminaries (continued)	40
B.1 Commitment Schemes and Security Definitions	41
B.2 Pseudo-Oracles	42
B.3 (Long-Term) Universal Composability	43
C Analysis of the Committed-Value Oracle \mathcal{O}_{CCA} (continued)	45
C.1 The Robust Extraction Lemma from [Goy+15]	45
C.2 Substitution rules	46
C.3 Asides	48
D Framework and Notion	49
D.1 Changes to the Framework	51
D.2 Protocol Emulation	52
D.3 Properties	53

E	Proof of Theorem 5	66
F	Applications	70
	F.1 Zero-Knowledge and Commit-and-Prove	70
	F.2 Oblivious Transfer with Long-term Security for One Party	71

A A Short Introduction to (G)UC

In the following, we give a brief overview of UC and GUC security. The following is adapted from [Bro+21]. For detailed introductions, see [Can01; Can+07].

In the (G)UC framework, security is defined by the indistinguishability of two experiments: the *ideal experiment* and the *real experiment*. In the ideal experiment, the task at hand is carried out by dummy parties with the help of an ideal incorruptible entity—called the ideal functionality \mathcal{F} . In the real experiment, the parties execute a protocol π in order to solve the prescribed task themselves. A protocol π is said to be a (secure) *realization* of \mathcal{F} if no PPT machine \mathcal{Z} , called the *environment*, can distinguish between these two experiments. In contrast to previous simulation-based notions, indistinguishability must not only hold after the protocol execution has completed, but even if the environment \mathcal{Z} —acting as the *interactive* distinguisher—takes part in the experiment, orchestrates all adversarial attacks, gives input to the parties running the challenge protocol, receives the parties’ output and observes the communication during the whole protocol execution.

UC Framework Conventions. In the (G)UC framework, each party is identified by its party identifier (PID) pid which is unique to the party and is the UC equivalent of the physical identity of this party. A party runs a protocol π is called the *main party* of this instance of π . A subsidiary and its parent use their *input/subroutine output* tape to communicate with each other. The set of machines taking part in the same protocol but for different parties communicate through their *incoming message* tapes. An instance of a protocol is identified by its session identifier (SID) sid . All machines taking part in the same protocol instance share the same SID. A specific machine is identified by unique its ID $\text{id} = (\text{pid}, \text{sid})$.

The (Dummy) Adversary. The adversary \mathcal{A} is instructed by \mathcal{Z} and represents \mathcal{Z} ’s interface to the network. To this end, all messages from any party to a party that has a different main party and that are intended to be written to an *incoming message* tape are copied to the adversary. The adversary can process the message arbitrarily. The adversary may decide to deliver the message (by writing the message on its own outgoing message tape), postpone or completely suppress the message, inject new messages or alter messages in any way including the recipient and/or alleged sender.

\mathcal{Z} may also instruct \mathcal{A} to corrupt a party. In this case, \mathcal{A} takes over the position of the corrupted party, reports its internal state to \mathcal{Z} and from then on may arbitrarily deviate from the protocol in the name of the corrupted party as

requested by \mathcal{Z} . This means whenever the corrupted machine would have been activated (even due to subroutine output), the adversary gets activated with the same input.

A special case for the adversary is the so-called *dummy adversary* that reports all received messages to the environment and delivers all messages coming from the environment. It can be shown that the dummy adversary is *complete*, i.e. that if a simulator for the dummy adversary exists, then there also exists a simulator for any other adversary.

Ideal Functionalities and the Ideal Protocol. An ideal functionality \mathcal{F} is a special type of entity whose instances bear a SID but no PID. Hence, it is an exception to the aforementioned identification scheme. Input to and subroutine output from \mathcal{F} is performed through dummy parties. Dummy parties merely forward their input to the input tape of \mathcal{F} and subroutine output from \mathcal{F} to their own outgoing message tape. They share the same SID as \mathcal{F} , but additionally have individual party identifiers (PIDs) as if they were the actual main parties of a (real) protocol. The ideal functionality \mathcal{F} is simultaneously a subroutine for each dummy party and conducts the prescribed task. $\text{IDEAL}(\mathcal{F})$ is called the (*ideal*) *protocol* for \mathcal{F} and denotes the set of \mathcal{F} together with its dummy parties.

The UC Experiment. Let π be a protocol, \mathcal{Z} an environment and \mathcal{A} an adversary. The UC experiment, denoted by $\text{Exec}_{\pi, \mathcal{A}, \mathcal{Z}}(n, a)$, initially activates the environment \mathcal{Z} with security parameter 1^n and input $a \in \{0, 1\}^*$. The first machine that is invoked by \mathcal{Z} is the adversary \mathcal{A} . All other parties invoked by \mathcal{Z} are set to be main parties of the challenge protocol π . \mathcal{Z} freely chooses their input, their PIDs and the SID of the challenge protocol. The experiment is executed as outlined above.

Definition of Security. Let π, ϕ be protocols. π *emulates* ϕ in the UC framework, denoted by $\pi \geq \phi$, if for every PPT adversary \mathcal{A} there is a PPT adversary \mathcal{S} such that for every PPT environment \mathcal{Z} there is a negligible function negl such that for all $n \in \mathbb{N}, a \in \{0, 1\}^*$ it holds that

$$|\Pr[\text{Exec}(\pi, \mathcal{A}, \mathcal{Z})(n, a) = 1] - \Pr[\text{Exec}(\phi, \mathcal{S}, \mathcal{Z})(n, a) = 1]| \leq \text{negl}(n),$$

where $\text{Exec}(\pi, \mathcal{A}, \mathcal{Z})(n, a)$ denotes the random variable for the environment \mathcal{Z} 's output in the UC execution experiment with protocol π and adversary \mathcal{A} on input a and security parameter n .

The simulator \mathcal{S} mimics the adversarial behavior to the environment as if this were the real experiment with real parties carrying out the real protocol with real π -messages. Moreover, \mathcal{S} must come up with a convincing internal state upon corrupted parties, consistent with the simulated protocol execution up to this point (dummy parties do not have an internal state).

Protocol Composition. UC security is closed under protocol composition: Let π, ϕ , be subroutine-respecting PPT protocols¹² and let ρ be an PPT arbitrary protocol. Then,

$$\pi \geq \phi \implies \rho^{\phi \rightarrow \pi} \geq \rho$$

where $\rho^{\phi \rightarrow \pi}$ is identical to ρ , except that sub-protocol instances of ϕ are replaced by instances of π .

The GUC Framework and GUC Security. Being a generalization of UC security, Generalized UC (GUC) security [Can+07] captures the case of *global subroutines*. Typical examples of global subroutines are ideal functionalities that are used by multiple protocols. In UC security, this is not possible as protocols accessing such a global functionality are not subroutine-respecting. Like UC security, GUC security features a general composition theorem and shares its properties like the completeness of the dummy adversary. For the formal definitions, please see [Can+07].

A.1 Important Ideal Functionalities

In the following, we give definitions of important ideal functionalities.

We start with the ideal functionality for commitments \mathcal{F}_{COM} , which allows a party C to commit to a bit b and to unveil it later on.

Definition 12 (The Ideal Functionality \mathcal{F}_{COM} , adapted from [CF01]). \mathcal{F}_{COM} proceeds as follows, running with a committer C and a receiver P_R and an adversary \mathcal{S} .

1. Upon receiving an input (`commit`, sid, b) from C, where $b \in \{0, 1\}$, record the value b and generate a public delayed output (`committed`, sid) to R.
2. Upon receiving an input (`unveil`, sid) from C, generate a public delayed output (`unveil`, sid, b) to R.

The ideal functionality \mathcal{F}_{CRS} models a common reference string that is accessible by all parties.

Definition 13 (The Ideal Functionality \mathcal{F}_{CRS} , adapted from [CF01]). \mathcal{F}_{CRS} proceeds as follows, when parameterized with a distribution D .

1. When activated for the first time on input (`value`, sid) by a party P , choose a value $d \leftarrow D$ and generate a public delayed out (`value`, sid, d) for P . Answer subsequent `value` inputs from parties P' by generating a public delayed output (`value`, sid, d) for P' .

The ideal functionality for zero-knowledge allows a prover P to prove the validity of a statement x to a verifier V .

Definition 14 (The Ideal Functionality \mathcal{F}_{ZK} , adapted from [Can+02]). \mathcal{F}_{ZK} proceeds as follows, running with a prover P , a verifier V and an adversary \mathcal{S} , and parameterized with a relation R :

¹² Very informally, a protocol π is subroutine-respecting if a machine μ of π does not communicate with a machine μ' that is not part of π or of a sub-protocol of π .

- Upon receiving $(\text{ZK} - \text{prover}, \text{sid}, x, w)$ from P , do: If $R(x, w) = 1$, generate a public delayed output $(\text{ZK} - \text{proof}, \text{sid}, x)$ to V and halt. Otherwise, halt without output.

The ideal functionality for commit-and-prove \mathcal{F}_{ZK} combines the functionalities \mathcal{F}_{COM} and \mathcal{F}_{ZK} , allowing a prover to repeatedly commit to values and to prove statements, using the committed values as witnesses.

Definition 15 (Ideal Functionality for Commit-and-Prove \mathcal{F}_{CP} , adapted from [Can+02]). \mathcal{F}_{CP} proceeds as follows, running with a committer C , a receiver V and an adversary \mathcal{S} , and is parameterized by a security parameter κ and a relation R :

- *Commit phase:* Upon receiving an input $(\text{commit}, \text{sid}, w)$ from C where $w \in \{0, 1\}^\kappa$, append the value w to the list \bar{w} and generate a public delayed output $(\text{receipt}, \text{sid})$ to V . (Initially, the list \bar{w} is empty.)
- *Prove phase:* Upon receiving an input $(\text{CP} - \text{prover}, \text{sid}, x)$ from C , where $x \in \{0, 1\}^{\text{poly}(\kappa)}$, compute $R(x, \bar{w})$: If $R(x, \bar{w}) = 1$, generate a public delayed output $(\text{CP} - \text{proof}, \text{sid}, x)$. Otherwise, ignore the input.

B Preliminaries (continued)

We write $[n] := \{1, \dots, n\}$, where $n \in \mathbb{N}$. For a probabilistic machine \mathcal{A} , we write $\mathcal{A}(x; r)$ for executing \mathcal{A} on input x with random tape r , and $\mathcal{A}(x)$ for (implicitly) choosing uniform r and executing $\mathcal{A}(x; r)$. We write $a \leftarrow \mathcal{A}(x)$ for the (probabilistic) output a of $\mathcal{A}(x)$, and $a \xleftarrow{\$} S$ for sampling a uniformly random element from a set S .

Definition 16 (k -round protocol). Let \mathcal{A} and \mathcal{B} be interactive (PPT) algorithms. A round in $\langle \mathcal{B}, \mathcal{A} \rangle$ is defined as one message sent either from \mathcal{A} to \mathcal{B} or from \mathcal{B} to \mathcal{A} .

If $\langle \mathcal{B}, \mathcal{A} \rangle$ has at most $k = k(\kappa)$ rounds, we say that it is a k -round protocol.

Definition 17 (Indistinguishable w.r.t. rewinding). Two (interactive oracle) algorithms $\mathcal{A}_0, \mathcal{A}_1$ are perfectly (resp. statistically) indistinguishable w.r.t. rewinding, if for any unbounded distinguisher \mathcal{D} which gets black-box (rewinding) access to \mathcal{A}_0 or \mathcal{A}_1 the distinguishing advantage is 0 (resp. negligible). Formally, black-box (rewinding) access to \mathcal{A}_b is defined by access to the next-message function of \mathcal{A}_b (with uniformly sampled and fixed random tape). In particular, \mathcal{D} learns all messages \mathcal{A}_b would send (including oracle queries) and responds in place of all communication partners (including oracles).

Corollary 2. Let \mathcal{A}_0 and \mathcal{A}_1 be oracle-algorithms and suppose they are perfectly indistinguishable w.r.t. rewinding. Then $\mathcal{A}_0^\mathcal{O}$ and $\mathcal{A}_1^\mathcal{O}$ are again perfectly indistinguishable w.r.t. rewinding for any (potentially unbounded) oracle \mathcal{O} .

Some (machine) modelling details Composition and interaction of machines can be specified very abstractly [MR11] or very concretely [Can01]. We take a middle ground: For concreteness, we assume that “direct interfacing” with machines such as oracles happens through a single external message tape, where a sender sends a message to a receiver by writing its own address, the receiver’s address, and the message on the tape. Moreover, there is some mechanism which ensures that only admissible messages are allowed (e.g., because machine \mathcal{B} may not have access to oracle \mathcal{O} of $\mathcal{A}^{\mathcal{O}}$). This modelling works nicely with black-box access to an interactive machine, especially since we can also interpret inputs and outputs of machine as special addresses, providing a uniform mechanism for modelling (non-interactive) algorithms with or without input and/or output. Moreover, in order to consider composed machines, say $\mathcal{A}^{\mathcal{O}}$, or $\langle \mathcal{B}, \mathcal{A} \rangle$, or $\langle \mathcal{B}, \mathcal{A}^{\mathcal{O}} \rangle$ as a single entity, we allow a machine to have multiple addresses. In case of $\langle \mathcal{B}, \mathcal{A} \rangle$, the addresses of \mathcal{B} and \mathcal{A} .

B.1 Commitment Schemes and Security Definitions

Definition 18 (Commitment Scheme). A commitment scheme (with setup), non-interactive unveil phase and message space \mathcal{M} is a tuple $(\text{Setup}, \mathcal{C}, \mathcal{R})$, where

1. **Setup** is a PPT algorithm which on input (1^κ) outputs a commitment key ck .
2. $\langle \mathcal{C}, \mathcal{R} \rangle$ is an interactive protocol with PPT machines.
3. The protocol has two phases: a commit phase and an unveil phase. In both phases, \mathcal{C} and \mathcal{R} receive common input $(1^\kappa, \text{ck})$, where κ is a security parameter 1^κ , and ck the commitment key. \mathcal{C} additionally receives a private input $v \in \mathcal{M}$ to be committed.
4. The commit phase results in a joint output c , called the commitment, a private output d for \mathcal{C} , called the decommitment. Without loss of generality, c can be the full transcript of the interaction between \mathcal{C} and \mathcal{R} .
5. In the unveil phase, committer \mathcal{C} sends the pair (v, d) to the receiver \mathcal{R} , and decides to accept or reject the decommitment (c, v, d) deterministically. We let OPEN denote the function that verifies the validity of (v, d) w.r.t. ck ; the receiver accepts (v, d) if $\text{OPEN}(\text{ck}, c, v, d) = 1$, and rejects otherwise.

If \mathcal{C} and \mathcal{R} do not deviate from the protocol, then \mathcal{R} should accept (with probability 1) during the unveil phase, where the probability is over the randomness used to generate ck , the randomness of \mathcal{C} and the randomness of \mathcal{R} . Moreover, we assume that \mathcal{M} is efficiently recognizable and \mathcal{R} rejects a decommitment if $v \notin \mathcal{M}$.

Standard commitments

Definition 19 (Binding). Let COM be a commitment scheme with message space \mathcal{M} , and let \mathcal{A} be a malicious committer. Let $\text{Exp}_{\mathcal{A}, \text{COM}}^{\text{bind}}(\kappa, z)$ be the output of the following experiment:

1. Run the malicious committer \mathcal{A} on input $(1^\kappa, z)$.
2. The experiment generates $\text{ck} \leftarrow \text{Setup}(1^\kappa)$.
3. \mathcal{A} gets ck as input and can engage in a commit phase with an honest receiver \mathcal{R} on common input (1^κ) . Let c denote the commitment.

4. After the commitment phase finishes, \mathcal{A} must provide two decommitments (v_0, d_0) and (v_1, d_1) .

5. \mathcal{A} wins the game if both of the following hold:

(a) $\text{OPEN}(\text{ck}, c, v_0, d_0) = \text{OPEN}(\text{ck}, c, v_1, d_1)$ and $v_0, v_1 \in \mathcal{M}$ and $v_0 \neq v_1$.

We define the advantage $\text{Adv}_{\mathcal{A}, \text{COM}}^{\text{bind}}(\kappa, z)$ as the probability that \mathcal{A} wins. A commitment scheme is computationally binding if for any PPT adversary \mathcal{A} , there exists a negligible function negl such that for all $\kappa \in \mathbb{N}$ and all $z \in \{0, 1\}^*$, the advantage of \mathcal{A} is negligible.

Now, we define the trapdoor property. The notion provides strong and statistical guarantees.

Definition 20 (Trapdoor Commitment Scheme). Let $(\text{Setup}, \text{C}, \text{R})$ be a commitment scheme with message space \mathcal{M} , and let $(\text{TSetup}, \text{C}_{\text{trap}})$ be algorithms which can be used in place of (Setup, C) . Then $\text{TRAPCOM} = (\text{Setup}, \text{C}, \text{R}, \text{TSetup}, \text{C}_{\text{trap}})$ is called trapdoor w.r.t. if

- $\langle \text{C}, \text{R} \rangle$ and $\langle \text{C}_{\text{trap}}, \text{R} \rangle$ are commitment schemes with message space \mathcal{M} , and
- for all PPT adversaries \mathcal{A} , it holds that

$$\begin{aligned} & \{\text{Exp}_{\mathcal{A}, \text{TRAPCOM}}^{\text{TDC}}(\kappa, 0, z)\}_{\kappa \in \mathbb{N}, z \in \{0, 1\}^*} \\ & \stackrel{s}{\approx} \{\text{Exp}_{\mathcal{A}, \text{TRAPCOM}}^{\text{TDC}}(\kappa, 1, z)\}_{\kappa \in \mathbb{N}, z \in \{0, 1\}^*} \end{aligned}$$

that is, the ensembles are statistically indistinguishable.

The experiment $\text{Exp}_{\mathcal{A}, \text{TRAPCOM}}^{\text{TDC}}(\kappa, b, z)$ is defined as follows:

1. Run $\mathcal{A}(1^\kappa, z)$ where \mathcal{A} interacts with the game \mathcal{G} as follows.
2. First, \mathcal{A} sends **(Setup)**. If $b = 0$, set $\text{ck} \leftarrow \text{Setup}(1^\kappa)$. Otherwise, set $(\text{ck}, \text{td}) \leftarrow \text{TSetup}(1^\kappa)$. The experiment sends ck to \mathcal{A} .
3. By sending **(Start, v)** to \mathcal{G} , \mathcal{A} starts the commit phase of TRAPCOM , acting as receiver. If $b = 0$, the experiment runs the code of the honest committer C on input $(1^\kappa, \text{ck}, v)$. If $b = 1$, the experiment runs the code of the trapdoor committer C_{trap} on input $(1^\kappa, \text{ck}, |v|, \text{td})$.
4. After the commit phase has finished, wait for a message **(Unveil)** and perform the unveil phase. If $b = 1$, the trapdoor committer receives v as additional private input.
5. The experiment outputs the view of \mathcal{A} .

Remark 7 (Multi-challenge experiments). In the multi-challenge binding experiment, the malicious committer may concurrently engage with arbitrarily many honest receivers. It wins if it wins in any session. In the multi-challenge trapdoor experiment, the malicious receiver may concurrently engage with arbitrarily many committers.

By a standard hybrid argument, multi-challenge security follows from single-challenge security.

B.2 Pseudo-Oracles

Remark 8. Our definition of pseudo-oracle limits their power as much as possible: Their only advantage over oracles is that they can access their caller's view. This

is sufficient to handle rewinding-based setting. However, a natural alternative definition is more simulation-based: A pseudo-oracle \mathcal{O} might simply be an algorithm which is given full (black-box) access to its caller \mathcal{A} and *replaces its caller*; $\mathcal{A}^{\mathcal{O}}$ denotes the resulting algorithm. In a sense, “ $\mathcal{A}^{\mathcal{O}}$ ” now actually denotes $\mathcal{O}^{\mathcal{A}}$, since \mathcal{O} can freely act *in place of* \mathcal{A} . While this significantly broadens the scope of what is considered a pseudo-oracle, the extraordinary power \mathcal{O} wields over its caller make any reasoning about “ $\mathcal{A}^{\mathcal{O}}$ ” basically impossible. E.g. \mathcal{A} may query its oracle on input 1^κ , ignore the response and then output 0^κ . Our notion of pseudo-oracle can never affect the output of \mathcal{A} in $\mathcal{A}^{\mathcal{O}}$. However, a simulation-based pseudo-oracle as outlined above can replace the output of \mathcal{A} with whatever it wants. Hence, in order for these relaxed notions to behave oracle-like, one must impose additional properties.

In summary, we intentionally choose a notion of pseudo-oracle, whose sole advantage over a standard oracle is that it learns the caller’s view (and the caller does not need to pass its view explicitly, and thus cannot lie about it).

Remark 9 (Interaction vs. multiple (pseudo-)oracles). In Definition 5, we allow the pseudo-oracle \mathcal{O} access the combined view of $\langle \mathcal{B}, \mathcal{A} \rangle$ when composed as $\langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}}$. This is arguably a natural choice w.r.t. to pseudo-oracles and composition of interactive algorithms. However, we note that the “correct” behavior of pseudo-oracles is less clear if an algorithm \mathcal{A} has with multiple (pseudo-)oracles. For example consider an oracle algorithm \mathcal{A} requires oracle \mathcal{O}_1 and pseudo-oracle \mathcal{O}_2 . Does \mathcal{O}_2 have access to the view of \mathcal{O}_1 in $(\mathcal{A}^{\mathcal{O}_1})^{\mathcal{O}_2}$ or not? Since oracles usually encapsulate a “special power” which \mathcal{A} gets access to, the conservative choice is to give \mathcal{O}_2 only view $_{\mathcal{A}}$. In particular, for black-box \mathcal{O}_1 , it is not possible to rewind \mathcal{O}_1 . For black-box \mathcal{O}_2 , this conservative choice ensures that $(\mathcal{A}^{\mathcal{O}_1})^{\mathcal{O}_2}$ and $(\mathcal{A}^{\mathcal{O}_2})^{\mathcal{O}_1}$ behave identically.

B.3 (Long-Term) Universal Composability

In this section, we briefly recapitulate the notion of *long-term universal composability* [MU10]. For a general introduction to (G)UC security, see Appendix A.

Long-term universal composability extends the notion of universal composability to a setting where *all* hardness assumptions eventually lose their validity. Intuitively, this captures a setting where information about a protocol execution is stored for the future, when cryptographic hardness assumptions may be broken due to e.g. computational advances or better methods for cryptanalysis.

Formally, the long-term execution is defined like UC protocol emulation, with the exception that the environment outputs an arbitrary string of polynomial length instead of a single bit. Without loss of generality, we may assume that the environment outputs its view, which contains all its information about the protocol execution. As environment and adversary are PPT machines (like with UC security), hardness assumptions remain valid throughout the protocol execution.

We say that a protocol π long-term-emulates a protocol ϕ if for all PPT adversaries \mathcal{A} , there exists a PPT simulator \mathcal{S} such that for all PPT environments

\mathcal{Z} , the output of \mathcal{Z} in an execution with π and \mathcal{A} is *statistically indistinguishable* from the output of \mathcal{Z} in an execution with ϕ and \mathcal{S} . Here, the requirement of statistical indistinguishability captures that hardness assumption no longer hold.

Long-term security features the same properties as UC security like completeness of the dummy adversary, transitivity and universal composability. For a formal treatment, see [MU10].

Unfortunately, long-term security is subject to even stronger impossibility results than UC security. To this end, we first recall the definition of *long-term revealing* functionalities as introduced by Müller-Quade and Unruh [MU10].

Definition 21 (Long-term Revealing Functionality, adapted from [MU10]).

For a given protocol execution, let trans denote the transcript of all communication between a functionality \mathcal{F} and all other machines (including the adversary). Let $\text{trans} \setminus \mu$ denote the transcript of all communication between \mathcal{F} and all machines except the machine with the extended identity μ . We say a functionality \mathcal{F} is long-term revealing (LTR) for μ if in any execution, there exists a deterministic function f (not necessarily efficiently computable) such that with overwhelming probability, we have $\text{trans} = f(\kappa, \text{trans} \setminus \mu)$, where $\kappa \in \mathcal{N}$ is the security parameter.

Intuitively, a functionality \mathcal{F} is long-term revealing for a party P if all communication between P and \mathcal{F} can be computed from all *other* communication with \mathcal{F} .

Remark 10. As we consider subroutine-respecting protocols only, the only parties communicating with a functionality \mathcal{F} are its *dummy parties*, which are part of the ideal protocol $\text{IDEAL}(\mathcal{F})$. Nevertheless, we will often say that \mathcal{F} is long-term-revealing for some party P of a protocol π which is in the \mathcal{F} -hybrid model, instead of referring to the appropriate dummy party of P .

Many widely-used and natural ideal functionalities are long-term revealing:

Lemma 2 (Examples for Long-term Revealing Functionalities, adapted from [MU10]). *Coin toss (\mathcal{F}_{CT}) and CRS (\mathcal{F}_{CRS}) with any distribution D are LTR for all parties. Commitment (\mathcal{F}_{COM}) and ZK (\mathcal{F}_{ZK}) are LTR for the recipient/verifier. If G is a key generation algorithm, such that the secret key depends deterministically on the public key (e.g. for RSA, ElGamal), the PKI \mathcal{F}_{KRK} parameterized with G is LTR for all parties registered with \mathcal{F}_{KRK} .*

Functionalities that are long-term-revealing for the committer (e.g. the ones in Lemma 2) cannot be used to long-term-UC-realize the ideal functionality for commitments \mathcal{F}_{COM} [MU10]. This stands in stark contrast to the well-known feasibility results of UC security, e.g. the possibility to construct a UC-secure commitment scheme (solely) in the \mathcal{F}_{CRS} -hybrid model [CF01]. Given that statistically hiding UC-secure commitment schemes can be constructed from standard assumptions [DN02], this impossibility result seems surprising. However, the statistically hiding property of e.g. [DN02] is only guaranteed in the *real* execution. In the *ideal* execution, the protocol may be computationally hiding

only (in order to allow straight-line extraction), incurring a large statistical distance between the executions. A similar argument holds for the OT protocol of Peikert, Vaikuntanathan, and Waters [PVW08], which can be instantiated to provide statistical security for one party at the expense of composability. For a discussion, see Appendix F.2.

To construct long-term-secure and composable commitment schemes, stronger setups such as a signature card [MU10] or a *physically unclonable function* (PUF) and a CRS [Mag+22] are necessary. One can also use protocols with statistical UC security. If more than 2/3 of the parties are honest (in case of malicious security), such protocols (e.g. [AL17]) can be constructed from ideal secure channels only. If no honest majority is available, protocols such as [DKM11] again require strong setups.

C Analysis of the Committed-Value Oracle \mathcal{O}_{CCA} (continued)

In this section, we recall the robust extraction lemma from [Goy+15], presented in our setting. Then we discuss a useful substitution rules and further asides.

C.1 The Robust Extraction Lemma from [Goy+15]

We recall the robust extraction lemma of [Goy+15].

Lemma 3 (Robust concurrent extraction, adapted from [Goy+15]). *Let COM' be the base commitment used in the PRS commitment and suppose that COM' has stateless receiver. Let ℓ be the rounds of the PRS preamble. Let \mathcal{E} be a black-box extractor with extraction based on the rewinding schedule `recurse` with extraction method `extract`. Let \mathcal{A} be a (not necessarily efficient) well-behaved adversary which expects access to a PRS extraction oracle. Let $M = 2^m$ be a bound on the maximal number of messages sent by \mathcal{A} , and let k bound the maximal number of $(\text{ExtSend}_i, m)$ messages of \mathcal{A} .*

1. **Extraction failure.** *Let E_{ExtFail} be the event that in the execution $\mathcal{E}^{\mathcal{A}}$, the extraction returned `ExtFail`. Then*

$$\Pr[E_{\text{ExtFail}}] \leq 2^{-\ell + (k+2)\log(M)} + M^2/|\mathcal{C}|$$

2. **Extraction efficiency.** *The number of oracle calls to \mathcal{A} by \mathcal{E} is bounded by M^2 . Aside from that, \mathcal{E} emulates the honest PRS receiver and does some bookkeeping. Thus, if \mathcal{A} is PPT, then asymptotically \mathcal{E} runs in time roughly $M(\kappa)^2 \text{poly}(\kappa)$ where $\text{poly}(\kappa)$ is the worst-case run-time of \mathcal{A} plus the PRS receiver and bookkeeping overhead per message.*
3. **Validity constraint (on the main thread).** *Let B be the event that in an execution, in some session s on the main thread the value $v \neq v_s$ is opened in the decommitment phase (and $v \neq \perp$), where v_s is the extracted value. Then*

$$\Pr[B] \leq \frac{1}{M \cdot \ell \cdot 2^\kappa} \text{Adv}_{\text{COM}', \mathcal{B}}^{\text{bind}}(\kappa)$$

where the adversary \mathcal{B} has runtime roughly that of \mathcal{E} applied to \mathcal{A} .

Proof. The extraction failure probability follows from the proof of [Goy+15, Lemma 1]. The efficiency can be derived from `recurse` directly (and is also part of [Goy+15, Lemma 1]). In both cases, our expression differs slightly, since we use M , an upper bound on the number of messages, instead of T , an upper bound on the number of sessions.

The validity constraint follows by a straightforward reduction, namely, guess the (first) session s^* , the (first) slot ℓ^* , and the index (i^*, b^*) of a commitment which is broken, and embed the (external) receiver from the binding game on the main thread. Observe that this is possible, because the base commitment is stateless by assumption. Thus, look-ahead threads can perfectly simulate embedded (honest) receiver as well.¹³ Recall that the base commitment scheme has non-interactive decommitment by assumption. Moreover, if the guess was correct, the extractor finds a valid *decommitment* d' (for value v_s) of the commitment in session s^* , slot ℓ^* and index (i^*, b^*) and the base decommitment d (for session s^* , slot ℓ^* , index (i^*, b^*)) unveiled later by \mathcal{A} is to a different value $v \neq v_s$ (and $v \neq \perp$). Thus, d' and d constitute a binding break, and the reduction adversary \mathcal{B} wins the binding game. \square

Remark 11. We remark that the loss factor $1/(M \cdot \ell \cdot 2\kappa)$, could be replaced by $1/(T \cdot \ell \cdot 2\kappa)$, where T is the maximal number of sessions opened by \mathcal{A} on the main thread. Or, one could strengthen validity to all threads; this increases the loss to $1/(M^2 \cdot \ell \cdot 2\kappa)$, since some sessions may exist in look-ahead threads only.

C.2 Substitution rules

Oftentimes, one wants to modify some game by moving some computation into or out of an oracle, e.g. the game may compute encryptions itself or query the oracle instead. For ordinary oracles, such changes are often trivially justified. With pseudo-oracles, the same problems as with composition-order invariance resurface. Thus, we have to establish substitution rules explicitly. With our committed-value oracle \mathcal{O}_{CCA} , the substitution rule of interest allows to move an honest receiver session into the \mathcal{O}_{CCA} oracle, or a session out of the \mathcal{O}_{CCA} oracle *provided that the extracted committed-value is ignored*.

This intuition can be formalized as follows: Let W_b for $b \in \{0, 1\}$ be a wrapper for \mathcal{O}_{CCA} and R , such that

- To start a new session, W_b expects an additional bit $e \in \{0, 1\}$ as input, which indicates whether the session’s committed-value will be extracted and returned (upon completion of the commitment phase) as in \mathcal{O}_{CCA} , or whether it will be ignored (i.e., replaced by \top).
- W_0 forwards everything to \mathcal{O}_{CCA} .
- W_1 forwards only sessions with $e = 1$ to \mathcal{O}_{CCA} , and runs R for $e = 0$.

¹³ In Lemma 1, it is described in more detail how to embed the reduction so that the PRS analysis still applies. Statelessness is used in to ensure look-ahead threads can continue the challenge receiver’s interaction. In [Goy+15], stateless receivers are not explicitly required for the validity constraint. See Remark 14 for a discussion.

By an argument similar to k -robust composition-order invariance, one obtains a k -robust substitution rule, which asserts that $\langle \mathcal{B}, \mathcal{A}^{W_0^{\mathcal{O}_{CCA}}} \rangle \stackrel{s}{\approx} \langle \mathcal{B}, \mathcal{A}^{W_1^{\mathcal{O}_{CCA}}} \rangle$. Note that rewriting a game so as to introduce or remove W_0 (resp. W_1) can be justified by black-boxness of \mathcal{O}_{CCA} , cf. Corollary 1.

Lemma 4. *Let \mathcal{B} , $W_0^{\mathcal{O}_{CCA}}$ and $W_1^{\mathcal{O}_{CCA}}$ as described above and suppose that $\langle \mathcal{B}, \mathcal{A} \rangle$ has at most k rounds. Suppose that COM' has a stateless receiver. Suppose $M = 2^m$ is an upper bound on the number of messages \mathcal{A} to the PRS oracle or to \mathcal{B} . Let T be an upper bound of the number of sessions started by \mathcal{A} on the main thread. Define the random variables*

- $\text{out}_1(\kappa, x, y, z)$ as $\text{out}_{\mathcal{B}, \mathcal{A}}(\mathcal{B}(x), \mathcal{A}^{W_0^{\mathcal{O}_{CCA}}}(y))(1^\kappa, z)$, and
- $\text{out}_2(\kappa, x, y, z)$ as $\text{out}_{\mathcal{B}, \mathcal{A}}(\mathcal{B}(x), \mathcal{A}^{W_1^{\mathcal{O}_{CCA}}}(y))(1^\kappa, z)$.

Then there exists an adversary $\mathcal{A}_{\text{COM}'}$ against multi-binding (Remark 7) with run-time roughly upper bounded by the maximal runtime of $\langle \mathcal{B}, \mathcal{E}^{A^{W_b^{\mathcal{O}_{CCA}}}} \rangle$ (for $b = 0$ or 1) (cf. Lemma 3) in expectation. In particular, if $W_0, W_1, \mathcal{B}, \mathcal{A}$ are PPT, so is $\mathcal{A}_{\text{COM}'}$ (as an oracle-algorithm). such that for all $\kappa \in \mathbb{N}$ and all $x, y, z \in \{0, 1\}^$, it holds that*

$$\Delta(\text{out}_1(\kappa, x, y, z), \text{out}_2(\kappa, x, y, z)) \leq 2 \cdot (2^{\ell - (k_{\text{ass}} + k + 2) \log(M)} + M^2 / |\mathcal{C}|) + 2^{-\kappa} + \frac{1}{T \cdot \text{poly}} \cdot \text{Adv}_{\text{COM}', \mathcal{A}_{\text{COM}'}}^{\text{bind}}(\kappa, z).$$

where $\text{poly}(\kappa) = \text{poly}_{\text{AoK}}(\kappa) + \kappa \cdot \ell(\kappa)$ and poly_{AoK} is a bound on the number of commitments made during in the AoK step, as in Lemma 1.

Proof (Proof sketch). The argument is similar to k -robust composition-order invariance, Lemma 1. Again, one fixes the randomness of \mathcal{B} and \mathcal{A} . Instead of “matching” the randomness of the main threads of \mathcal{O}_{CCA} in two different executions, as in Lemma 1, one (fixes and) “matches” the randomness of W_b and \mathcal{O}_{CCA} . (Recall that the randomness of \mathcal{O}_{CCA} is can be structured suitably to simplify this matching, cf. Remark 4.) That is,

- $W_0^{\mathcal{O}_{CCA}}$ simply runs everything through \mathcal{O}_{CCA} . Let r' be the randomness of the main thread of \mathcal{O}_{CCA} , i.e. the challenges sent by \mathcal{O}_{CCA} .
- $W_1^{\mathcal{O}_{CCA}}$ runs sessions with $e = 1$ through \mathcal{O}_{CCA} and those with $e = 0$ are emulated by W_1 itself. Let $r'_{\mathcal{O}_{CCA}}$ be the randomness in the main session of \mathcal{O}_{CCA} and r'_W be the randomness in the sessions run by W_1 , i.e. the challenges sent by \mathcal{O}_{CCA} resp. W_1 .

Observe that there is an obvious mapping between r' and $(r'_{\mathcal{O}_{CCA}}, r'_W)$. Moreover, both specify behavior on the main thread completely *as long as extracted values on the main threads do not diverge*,¹⁴ as in Lemma 1. Following the proof of Lemma 1, we get a statistical bound on divergence plus a reduction to the

¹⁴ This mapping is not strictly a bijection, since r' and $r'_{\mathcal{O}_{CCA}}$ already have the same size. However, the “actually used” prefixes of the main thread randomness r' and $(r'_{\mathcal{O}_{CCA}}, r'_W)$ are evidently in bijection. After the main thread terminates, the mapping is unspecified — but then it is also irrelevant for the output.

binding property of the base commitment COM' (which includes soundness of the AoK) which ensures that divergent extractions on the main thread happen with probability at most

$$2 \cdot (2^{-\ell + (k_{\text{ass}} + k + 2) \log(M)} + M^2 / |\mathcal{C}|) + 2^{-\kappa} + \frac{1}{T \cdot \text{poly}} \cdot \text{Adv}_{\text{COM}', \mathcal{A}_{\text{COM}'}}^{\text{bind}}(\kappa, z)$$

for a suitable (expected-time) adversary $\mathcal{A}_{\text{COM}'}$. Thus, the claim follows. \square

Remark 12. We formulated Lemma 4 with k -robustness for \mathcal{B} for convenience. As a corollary of composition-order invariance, \mathcal{B} could be introduced anyway. Yet, unlike composition-order invariance, no “break-points” change and thus, \mathcal{O}_{CCA} is essentially unaffected by \mathcal{B} . Thus, it may be possible to make Lemma 4 independent of k . For now, this setting appears to be of little interest.

C.3 Asides

Example 1 (PRS is not necessarily COI). The composition-order invariance PRS-commitments depend on their definition of `extract` (which, following [Goy+15], we left open in cases of ambiguities). If `extract` outputs the value of a random extracted slot, the following is an attack on COI: External algorithm \mathcal{B} does nothing, except acknowledge receipt of a message. The adversary \mathcal{A} runs a single PRS-commit to value 1 almost honestly, except in a random slot, where it commits to 0. Moreover, \mathcal{A} wraps that random slot in external messages to \mathcal{B} . Now in case $\langle \mathcal{B}, \mathcal{A}^{\text{CCA}} \rangle$, all extracted slots yield 1. In case $\langle \mathcal{B}, \mathcal{A}^{\text{CCA}} \rangle$, there is a non-negligible probability that the slot with 0 is extracted and outputted.

Small variations of this example show that it does not help to output \perp if not all extracted values are consistent, nor does a simple majority decision avoid an attack. Nevertheless, this does not rule out COI for a suitable `extract`.

Remark 13. It is not obvious how far the requirements in Lemma 1 could be relaxed, i.e. whether a reduction to binding is strictly necessary, or if it is possible to avoid it, and a similar result holds unconditionally.

Remark 14 (Necessity of special commitment schemes). While a stateless receiver of COM' may be traded for other (stronger) notions of binding in Lemma 1, it seems necessary to impose requirements beyond generic binding. Consider following pathological example: The receiver “protects” its messages by using a signature (or MAC) on the partial transcript and its response. The committer and receiver check these authentications, and halt if they are invalid. Clearly, the receiver is not stateless anymore. Moreover, suppose the commitment has many rounds, e.g. by adding dummy rounds. Now, embedding a binding challenge is not as simple anymore: The scheduling chosen by \mathcal{A} might require the reduction to continue a partially completed embedded *challenge commitment* in a look-ahead thread, but with different responses from \mathcal{A} (e.g. add some garbage (e.g. randomness or hash of the view) to make sure \mathcal{A} sends different messages with overwhelming probability). While continuing the receiver was trivial for stateless

receivers, now, the reduction has to break EUF-CMA security of the signature (or MAC) scheme. This seems to preclude simple (black-box) reductions.

We also note that similar problems apply when establishing the “validity constraint” for [Goy+15, Lemma 6]. The idea to circumvent problems by moving the binding challenger to the left side runs afoul to composition-order invariance. Indeed, examples which show that COI fails for PRS preamble extraction in general (e.g. Example 1), can be modified to apply adapt to this situation. Thus, some non-trivial justification (or the restriction of COM') is required for [Goy+15, Lemma 6] as well.

Remark 15 (Relation to [Goy+12, Lemma 6]). At first glance, Lemma 1 might be superfluous as the generalized robust concurrent extraction lemma in [Goy+12] could be used instead. However, we ran into some obstacles. Firstly, we failed to justify [Goy+12, Lemma 6] for general commitment schemes as noted in Remark 14. And secondly, there are unfortunate ambiguities in [Goy+12], so it is not clear if and how their generalized robust concurrent extraction lemma would apply. More precisely:

- The extractor \mathcal{E} in [Goy+12] merely interacts with the adversary. As such, it is impossible for \mathcal{E} to run the (rewinding-based) simulation for statistically hiding PRS preambles. To fix this, we view \mathcal{E} as a black-box pseudo-oracle.
- The formal statement, [Goy+12, Lemma 6], claims in constraint (b) that *for every statistically hiding preamble*, the extracted decommitment will coincide with a potential value unveiled by \mathcal{A} . This suggests that constraint (b) holds with probability 1, but evidently, it only holds by reduction to the binding property, so with overwhelming probability (at best). While missing in the statement of [Goy+12, Lemma 6], it is clearly explained before and after [Goy+12, Lemma 6]. Indeed, a proof sketch is given which hints at a reduction (which, as noted before, we could only justify for stateless commitment schemes).
- With the proposed corrections to the statement and the extractor (and assuming stateless commitment schemes), one observes that it is not (obviously) possible to swap out the external protocol Π from the rewinding (of the simulator \mathcal{S}) anymore, because the extractor \mathcal{E} (which is *not* straight-line anymore) acts exactly as \mathcal{S} for statistically hiding preambles, and thus also uses rewinding and is dependent on the external protocol Π .¹⁵

Thus, at least when the ambiguities are resolved as suggested, there is still a gap we have to fill for our proofs to work. This is addressed by Lemma 1.

D Framework and Notion

¹⁵ In the *non*-generalized robust extraction lemma [Goy+12, Lemma 1], \mathcal{E} is a normal oracle and extraction is straight-line. As such, it is trivial to see that “swapping out” which protocol parts are considered the external protocol does not affect $\text{REAL}_{\mathcal{E}, \Pi}^{\mathcal{A}}$. Consequently, “swapping out” the external protocol also works for simulations, simply by arguing through the extractor \mathcal{E} and indistinguishability of \mathcal{E} and the respective simulator \mathcal{S} .

This chapter contains the full treatment of framework and notion. Major changes and additions are highlighted in boxes. For the sake of a cohesive presentation, we re-state the other parts unmodified and unboxed.

With UC security and its variants, all entities keep their run-time complexity throughout the whole execution, making them unsuitable to analyze the security of protocols in a setting where cryptographic hardness assumptions may lose their validity.

In order to circumvent the impossibility results of Long-Term Security [MU10], we want to modify the protocol execution such that rewinding-based extraction of long-term hiding commitment schemes is possible in a way that preserves universal composability.

To this end, we take a route similar to [PS04; CLP10] and provide environment and adversary with an entity called the *helper* \mathcal{H} . This *efficient* helper is parameterized with an extractable commitment scheme COM and allows the rewinding-based extraction of instances of COM where the commitment is performed with \mathcal{H} as receiver.

Formally, our notion is cast in the GUC framework [Can+07], allowing both the use of the helper \mathcal{H} as well as other global ideal functionalities. We assume that the reader is familiar with the basic concepts of (G)UC security. For a short overview, see Appendix A. Due to space constraints, this chapter is a short version only. For the full treatment of framework and notion, see Appendix D.

Extracting (Statistically Hiding) Commitments. \mathcal{H} provides an oracle that allows the extraction of commitments (cf. Section 3.2), similar to a CCA oracle. This part is analogous to the helper of [CLP10], with the following differences: The helper of [CLP10] is able to extract statistically binding commitments by inefficient computations. In contrast, we want to consider commitments that are statistically hiding. Such commitments cannot be extracted by brute force, but require different techniques such as an appropriate setup allowing for straight-line extraction (see [MU10] for an example) or rewinding. More specifically, we adapt the rewinding-based extraction techniques of [Goy+15] to our setting, via pseudo-oracles and a suitably adapted analysis in Sections 3 and 4. We provide the helper with the views of all ITIs that may be affected by a performed rewinding.

While we do not (intend to) achieve composability in the plain model, the resulting security notion has properties and limitations similar to Angel-based security [PS04] or UC with super-polynomial helpers [CLP10], e.g. with respect to protocol reusability.

As we will later use commitment schemes in the \mathcal{F}_{CRS} -hybrid model, we have adapted the helper accordingly. When a corrupted party starts a new commitment session with \mathcal{H} , the committed-value oracle \mathcal{O}_{CCA} within \mathcal{H} honestly executes the CRS generation algorithm of the desired commitment scheme and \mathcal{H} returns the resulting CRS ck to the party initiating the session.

As the commitment key is generated honestly, it is guaranteed to be independent from all other commitment keys. Thus, a corrupted party cannot take a key ck' from another session (e.g. where the sender is honest) and have \mathcal{H}

extract commitments relative to ck . A similar policy is enforced in [CLP10] by the use of tags, which we omit as they are not necessary (with different sessions distinguished by their commitment key).

To establish meaningful properties, we require the commitment scheme to feature a committed-value oracle which is *black-box* (Definition 4) and *k-robust composition-order invariant and pseudo-PPT* (Definitions 5 and 6). This allows to a) import protocols with appropriate round complexity into our framework without loss of security due to the committed-value oracle and b) prove the security of protocols within our framework by reducing to security properties with a certain (bounded) round complexity. The robustness property guarantees that we can efficiently simulate the committed-value oracle without having to rewind the challenger in a reduction.

The helper \mathcal{H} is formally defined in Definition 9.

Definition 22 (The helper \mathcal{H}). \mathcal{H} is parameterized with 1. a security parameter $\kappa \in \mathbb{N}$, 2. auxiliary input z and 3. a commitment scheme COM with committed-value (pseudo-)oracle \mathcal{O}_{CCA} .

- Upon receiving an input $(\text{corrupt}, P_i, \text{sid})$ from the environment, record $(\text{corrupt}, P_i, \text{sid})$.
- Upon receiving an input $(\text{ext-init}, P_i, \text{sid}, k)$ from a corrupted party P_i in the protocol with SID sid : If there is a recorded session (P_i, sid, k) , ignore this message. Otherwise, initialize the k -th sub-session of (P_i, sid) with \mathcal{O}_{CCA} and receive a setup ck . Record session (P_i, sid, k) and return $(\text{setup}, \text{sid}, k, ck)$ to P_i .
- Upon receiving a message $(\text{ext-msg}, P_i, \text{sid}, k, m)$ from a corrupted party P_i in the protocol with SID sid : If there is no recorded session (P_i, sid, k) , ignore the message. Otherwise, give input (sid, k, m) to \mathcal{O}_{CCA} , possibly obtain a reply m' . If m' is a special message $(\text{End}, s, w_s, \text{view}_{R_s})$, return $(\text{ext-val}, P_i, \text{sid}, k, w_s)$ to P_i . Otherwise, return $(\text{ext-msg}, P_i, \text{sid}, k, m')$ to P_i .

In the following, we first introduce changes to the framework that are necessary to enable the extraction of statistically hiding commitments. Then, we adapt the definitions of protocol emulation from [CLP10; Can+07; MU10] and discuss the properties of the new notion.

D.1 Changes to the Framework.

Due to the technicalities of pseudo-oracles outlined in Section 3.1, we need to slightly adapt the model of execution, so that the pseudo-oracle (and implicitly the helper \mathcal{H}) have access to the view(s) of all (possibly dynamically created) instances of interactive Turing machines (ITIs).

Execution ITI. We assume that all ITIs, with the exception of \mathcal{H} , are emulated within a special “execution ITI” \mathcal{E} . In particular, \mathcal{E} executes the control function, the environment, the adversary as well as all other entities

like protocol parties and (global) ideal functionalities. \mathcal{E} also appropriately maps the communication between internally emulated ITIs and \mathcal{H} , which is not governed by the (G)UC control function anymore, but is subject to the usual mechanisms and rules of communication in the (G)UC framework. \mathcal{H} can then provide its internal execution of \mathcal{O}_{CCA} with the necessary view(s), cf. Definition 5. Clearly, the introduction of \mathcal{E} incurs only at most a polynomial overhead compared to an execution without \mathcal{E} , i.e. where all entities run on individual ITIs and \mathcal{H} is provided with the necessary randomness via some other mechanism.

Changes to the Execution Experiment. We modify the execution experiment to use \mathcal{E} as follows:

1. \mathcal{E} is the first ITI (initial ITI) to be invoked on input $(1^\kappa, z)$.
2. On invocation, \mathcal{E} immediately invokes \mathcal{H} , giving its code¹⁷ and input to \mathcal{H} as first input.
3. On its first activation, \mathcal{H} immediately activates \mathcal{E} again.
4. \mathcal{E} continues the internal execution of the (G)UC experiment, interacting with \mathcal{H} .
5. \mathcal{H} has read-only access to the random tape of \mathcal{E} .
6. Eventually, \mathcal{E} outputs what the internally executed environment outputs. The random variable $\text{Exec}(\pi, \mathcal{A}, \mathcal{Z})(\kappa, z)$ is re-defined accordingly.

As \mathcal{E} is merely a wrapper that does not affect the (G)UC execution it emulates in any way, we will ignore it from now on. In particular, we will adhere the usual conventions and notation.

We call the framework resulting from above modifications the *UC Security with Rewinding framework*.

D.2 Protocol Emulation

Before stating long-term protocol emulation, we provide the standard notion of computational protocol emulation adapted to our setting.

Definition 23 (UC Security with Rewinding Protocol Emulation). *Let π and ϕ be PPT protocols and let \mathcal{H} be the helper of Definition 9. We say that π Rewinding-UC-emulates ϕ if for all PPT adversaries \mathcal{A} , there exists a PPT simulator \mathcal{S} such that for all \mathcal{H} -aided¹⁸ balanced PPT environments \mathcal{Z} , there exists a negligible function negl such that for all $\kappa \in \mathbb{N}, z \in \{0, 1\}^*$ it holds that*

$$|\Pr[\text{Exec}(\pi, \mathcal{A}, \mathcal{Z})(\kappa, z) = 1] - \Pr[\text{Exec}(\phi, \mathcal{S}, \mathcal{Z})(\kappa, z) = 1]| \leq \text{negl}(\kappa)$$

¹⁷ Actually, it suffices to give \mathcal{H} black-box (rewinding) access to \mathcal{E} .

¹⁸ We restate the definition of \mathcal{H} -aided environments due to Canetti, Lin, and Pass [CLP16]: a) \mathcal{Z} invokes a single instance of \mathcal{H} immediately after invoking the adversary. b) As soon as a party (i.e. an ITI) P is corrupted (i.e. P receives a **corrupted** message), \mathcal{Z} lets \mathcal{H} know of this fact. \mathcal{H} interacts only with the environment, the adversary, and the corrupted parties.

If π Rewinding-UC-emulates ϕ , we write $\pi \geq_{\mathbb{R}} \phi$.

We adapt the notion of long-term protocol emulation in our framework in analogy to the established definition due to Müller-Quade and Unruh [MU10]. In contrast to standard UC emulation, long-term emulation allows the environment to output an arbitrary string of polynomial length and requires statistical indistinguishability of the resulting ensembles. Intuitively, this means that all (polynomial-time) hardness assumptions lose their validity after the protocol execution has finished.

To this end, let ExecS denote the random variable that is identically defined to Exec , except that the environment outputs an arbitrary string (of polynomial length).

Definition 24 (Long-term UC Protocol Emulation). *Let π and ϕ be PPT protocols and let \mathcal{H} be the helper of Definition 9. We say that π long-term-Rewinding-UC-emulates ϕ if for all PPT adversaries \mathcal{A} , there exists a PPT simulator \mathcal{S} such that for all \mathcal{H} -aided balanced PPT environments \mathcal{Z} , the ensembles $\{\text{ExecS}(\pi, \mathcal{A}, \mathcal{Z})(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^*}$ and $\{\text{ExecS}(\phi, \mathcal{S}, \mathcal{Z})(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^*}$ are statistically indistinguishable.*

If π long-term-Rewinding-UC-emulates ϕ , we write $\pi \geq_{\mathbb{R}}^{\text{lt}} \phi$. If π long-term-Rewinding-UC-emulates the ideal protocol of a functionality \mathcal{F} , then we say that π long-term-Rewinding-UC-realizes \mathcal{F} .

Remark 16. In contrast to the definition of long-term security in [MU10], the environment of Definition 11 has access to the helper \mathcal{H} , which provides a committed-value oracle that does not exist in the original definition.

It is easy to see that long-term emulation implies classical emulation:

Proposition 4 (Long-term Emulation Implies Classical Emulation).

Let π and ϕ be PPT protocols. If $\pi \geq_{\mathbb{R}}^{\text{lt}} \phi$, then $\pi \geq_{\mathbb{R}} \phi$ (for the same helper \mathcal{H}).

The proof of Proposition 4 is simple and we omit it.

D.3 Properties

We now discuss the properties of our notion, which are mostly similar to the properties of long-term security and UC security with super-polynomial helpers.

Proposition 5 (Completeness of the Dummy Adversary). *The dummy adversary is complete.*

The proof is identical to the one for UC security and we omit it.

Also, our notion is transitive.

Proposition 6 (Transitivity). *Let π_1, π_2, π_3 be PPT protocols. If π_1 (long-term-) Rewinding-UC-emulates π_2 and π_2 (long-term-) Rewinding-UC-emulates*

π_3 (for the same helper \mathcal{H}), then π_1 (long-term-) Rewinding-UC-emulates π_3 .

As the proof is very similar to the one for transitivity of GUC security, we omit it.

Like UC security with superpolynomial helpers [CLP10] and long-term security [MU10]), our notion is closed under general concurrent (i.e. universal) composition.

Theorem 6 (Composition Theorem). *Let ρ, π, ϕ be PPT protocols where π and ϕ subroutine-respecting. If π (long-term-) Rewinding-UC-emulates ϕ , then, $\rho^{\phi \rightarrow \pi}$ (long-term-) Rewinding-UC-emulates ρ .*

Again, the proof is very similar to the ones presented in [CLP10; Can+07; MU10] and we omit it.

UC Compatibility. When introducing a new security notion that features modular design, a natural question to ask is which existing protocols (that are secure according some other notion) can be reused.

Let π and ϕ be PPT protocols such that $\pi \geq_{\text{UC}} \phi$. Due to the helper \mathcal{H} , just as in [PS04; CLP10], we cannot hope that we can import an arbitrary UC protocol π securely, i.e. that $\pi \geq_{\text{UC}} \phi$ implies that $\pi \geq_{\text{R}} \phi$. This is because a Rewinding UC environment is more powerful than a normal UC environment due to the access to \mathcal{H} : The committed-value oracle of \mathcal{H} could invalidate assumptions made in the security proof.

Nevertheless, we can show the compatibility with UC security for large classes of protocols, namely those that have less than or equal to k rounds if the committed-value (pseudo-)oracle provided by \mathcal{H} is black-box (Definition 4) k -robust composition-order invariant (Definition 5) and pseudo-PPT (Definition 6). This criterion essentially is the same as in [CLP10], except for the additional requirements for the (pseudo-)oracle.

Before stating the theorem, we give a formal definition of k -round protocols. As the model of execution in Rewinding UC is different from stand-alone execution, we cannot simply reuse the stand-alone definition of k -round protocols (Definition 16).

As we eventually want to make use of the k -robust composition-order invariance (Definition 5), we need a definition of k -round protocols within Rewinding UC that is compatible with the stand-alone definition. In particular, this compatibility needs to hold in the case where protocol and (dummy) adversary are considered as a “left side” and everything else (i.e. environment and helper) as a “right side”.

The same is necessary to argue the compatibility with UC security in e.g. [CLP10]. Unfortunately, Canetti, Lin, and Pass [CLP10] do not give a definition of k -round UC protocols.²¹ We thus provide a possible definition here. We stress that any definition that works for [CLP10] works for our setting and vice versa.

Definition 25 (Protocol Round). Let π be a subroutine-respecting PPT protocol. We define a round of π to be one of the following:

1. Input given to a main party.
2. Subroutine output given by a main party.
3. Subroutine output given by a sub-party P (i.e. not a main party) triggered by input of a (sub-party) P' such that
 - P is a subsidiary of a main party M of π and
 - P' is a subsidiary of a main party $M' \neq M$ of π or the recipient of P 's output is not a subsidiary of M .
4. Messages between (sub-)parties of π and the adversary.
5. Messages between the adversary and an ideal functionality.

Remark 17. Item 3 captures messages sent from a sub-party μ , which is part of a larger protocol π , to a sub-party μ' (of π) via an ideal functionality (and the corresponding dummy parties), e.g. via $\mathcal{F}_{\text{AUTH}}$ or \mathcal{F}_{SMT} . However, “local” interactions with an ideal functionality, e.g. with a random oracle, are not counted as a round. Unless governed by Item 3, “immediate” communication within a protocol (i.e. through inputs and subroutine outputs) is not counted towards the number of protocol rounds, as it is not externally visible (unless a party is corrupted).

Definition 26 (k -round Protocol). Let π be a subroutine-respecting PPT protocol. We say that π is a k -round protocol if for all environments \mathcal{Z} and adversaries \mathcal{A} interacting with π , there exists

- for each (honest) main party P_i of π a bound n_i^I for the number of inputs for P_i ,
- for each (honest) main party P_i of π a bound n_i^O for the number of subroutine outputs by P_i ,
- for each (honest) main party P_i of π (and its sub-parties) a bound n_i for the number of rounds according to Items 3 and 4 in Definition 25,
- for each (honest) main party P_i of π (and its sub-parties) a bound n_i^A for the number of rounds for communication with the adversary according to Item 4 in Definition 25,
- a bound n_A for the communication with the adversary according to Item 5 in Definition 25 if π is the ideal protocol of \mathcal{F}

such that the number of rounds of π is bounded by $k = k(\kappa) = n_A + \sum_{P_i \in \mathcal{P}} n_i^I + n_i^O + n_i + n_i^A$. Here, \mathcal{P} denotes the set of main parties of π that may be jointly invoked.

Remark 18. Definitions 25 and 26 impose hard restrictions on the number of protocol rounds. For example, a (main) party P of a k -round two-party protocol will halt after receiving k messages from the adversary, regardless of whether these messages are valid in the context of π . This could be possibly modified to only count “valid” protocol messages at the sake of a more

complicated definition and the possible challenge to identify “valid” protocol messages.

Remark 19. Other definitions of protocol rounds and k -round protocols are conceivable. In particular, it may hold that protocols that are (informally) considered to be k -round are not so according to our definition. However, we believe that our notion is sufficiently general as it naturally covers many protocols (in particular if they are adapted to halt after a certain number of rounds).

Throughout the paper, we assume that protocols and functionalities with a bounded round complexity adhere to their natural bound of rounds, in particular counting bogus messages (from the adversary) towards the number of rounds.

Example 2. We analyze the round complexity of several protocols.

- $\text{IDEAL}(\mathcal{F}_{\text{COM}})$ where the commit and unveil phase are performed has eight rounds:
 - Two rounds for the input of the committer.
 - Two rounds for the output of the receiver.
 - Four rounds for the communication between \mathcal{F}_{COM} and the adversary (i.e. the delayed outputs).
- $\text{IDEAL}(\mathcal{F}_{\text{AUTH}})$ where the message is delivered has four rounds:
 - Two rounds for input and subroutine output.
 - Two rounds for the communication between $\mathcal{F}_{\text{AUTH}}$ and the adversary (i.e. the delayed output).
- Let π be the protocol that has two main parties P_1 and P_2 . P_1 accepts one input, invokes an instance of $\mathcal{F}_{\text{AUTH}}$ to send its input it to a (sub-party of) P_2 . Upon receiving subroutine output y from (its sub-party of) $\mathcal{F}_{\text{AUTH}}$, P_2 outputs y and both parties halt. π has five rounds:
 - Two rounds for input and subroutine output of P_1 and P_2 .
 - One round for the communication via $\mathcal{F}_{\text{AUTH}}$.
 - Two rounds for the communication between $\mathcal{F}_{\text{AUTH}}$ and the adversary (i.e. the delayed output).

We stress that, in order to be a k -round protocol according to Definition 26, the protocols above need not accept additional messages after k rounds have been performed in total. This may not be satisfied by the usual definitions of e.g. \mathcal{F}_{COM} or $\mathcal{F}_{\text{AUTH}}$.

Remark 20. Example 2 illustrates the number of rounds of a protocol π , according to Definition 26, may be smaller than the number of rounds of its building blocks. However, it is easy to see that an upper bound for the number of protocol rounds can be obtained by adding the number of rounds of π and its components.

Remark 21. Note that the numbers of rounds of a protocol π may change under composition, i.e. when a sub-protocol ϕ of π is replaced with a sub-protocol σ with a different number of rounds.

Remark 22. Often, we consider a protocol π that executes several instances of a sub-protocol ϕ *in parallel*, e.g. a commitment scheme. While we would like to count all messages of the l -th round of all m instances of ϕ executed in parallel as *one* round, this is technically incorrect: In the UC framework, only one **external write** instruction can be issued at the same time, formally leading to m rounds in π for every round of ϕ .

If the number of instances m executed in parallel is known at the time of invocation of ϕ , we can instead consider a wrapper protocol ϕ' that includes all m instances of ϕ and performs all communication in parallel. By modifying π to use instance of ϕ' instead of (several parallel instances of) ϕ , we obtain a protocol with the “correct” round complexity.

We will implicitly use this transformation in the following.

With the above definition at hand, we are ready to state the following theorem.

Theorem 7 (UC Compatibility). *Let \mathcal{H} be the helper that is parameterized with a commitment scheme COM that features an $O(k)$ -robust black-box composition-order invariant pseudo-PPT committed-value (pseudo-)oracle \mathcal{O}_{CCA} , where $k \in O(\text{poly}(\kappa))$. Let ϕ be a subroutine-respecting PPT protocol and let π be a subroutine-respecting PPT protocol with less than or equal to k rounds*

according to Definition 26

such that

- $\pi \geq_{\text{stat-UC}} \phi$. Then, $\pi \geq_{\text{R}}^{\text{lt}} \phi$.
- $\pi \geq_{\text{ltUC}} \phi$. Then, $\pi \geq_{\text{R}}^{\text{lt}} \phi$.
- $\pi \geq_{\text{UC}} \phi$. Then, $\pi \geq_{\text{R}} \phi$.

Here, $\geq_{\text{stat-UC}}$ denotes statistical UC emulation, \geq_{ltUC} denotes long-term emulation and \geq_{UC} denotes standard UC emulation.

Proof. We only prove the first part of Theorem 7, as the other parts are very similar. Let π be a subroutine-respecting PPT protocol with up to k rounds such that $\pi \geq_{\text{stat-UC}} \phi$. Let \mathcal{S} be the (PPT) simulator for the dummy adversary in the UC execution. We transform \mathcal{S} to a (presumptive) simulator \mathcal{S}' in the Rewinding UC execution. Namely, \mathcal{S}' is identical to \mathcal{S} but additionally handles messages between \mathcal{H} and corrupted parties like the dummy adversary. We recall that, according to the definition of statistical

²¹ Canetti, Lin, and Pass [CLP10] focus on constant-round protocols, which are usually easy to recognize. However, they state that their result can be extended to the general case.

UC security, the run-time of \mathcal{S} is polynomial in the run-time of the adversary it simulates. As we consider only polynomial-time adversaries, \mathcal{S}' is PPT.

First, we note that the round complexity between the environment and π and \mathcal{D} in an UC execution of π and \mathcal{D} is bounded by $O(k)$ if π is a k -round protocol according to Definition 26.

As π emulates ϕ , this also holds in the UC execution with ϕ and the simulator \mathcal{S} for the dummy adversary.

For the sake of contradiction, assume that $\pi \not\approx_{\mathbb{R}}^{\text{lt}} \phi$ for the dummy adversary and simulator \mathcal{S}' and some Rewinding UC environment \mathcal{Z} . Let D be an (unbounded) distinguisher that distinguishes with non-negligible advantage.

We construct an (unbounded) UC environment \mathcal{Z}' that uses the Rewinding UC environment \mathcal{Z} to break the statistical UC emulation of π and ϕ .

Let \mathcal{E} be the ITM defined in Appendix D.1 that interacts with \mathcal{H} . Clearly, \mathcal{E} is PPT. Let \mathcal{E}'_1 be identical to \mathcal{E} , but instead of interacting with \mathcal{H} , interact with a (pseudo-)oracle \mathcal{O}_{CCA} and adapt the communication as necessary. Using the black-box property of \mathcal{O}_{CCA} , we can establish that this change is perfectly indistinguishable.

We now “externalize” the challenge protocol and the dummy adversary resp. simulator and treat them as a left side. Note that this does not work directly for \mathcal{S}' , as \mathcal{S}' may perform queries to \mathcal{H} for the environment, leading to more than $O(k)$ rounds to be performed in the external interaction. However, we can avoid the problem in the following by differently handling these messages.

Towards this, we state and prove the following proposition.

Proposition 7. *Let π be a subroutine-respecting k -round PPT protocol according to Definition 26. Let \mathcal{H} be a helper. Let T_1 be the Turing machine comprised of (the honest parties of) π and an adversary \mathcal{D} with the interface of the dummy adversary and T_2 be the Turing machine comprised of the environment \mathcal{Z} , the helper \mathcal{H} as well as an adversary \mathcal{D}' that is defined as follows:*

- Messages from the environment to corrupted parties intended for \mathcal{H} (and vice versa) are forwarded between \mathcal{H} and \mathcal{Z} .

T_1 and T_2 communicate as follows:

- T_1 forwards messages from its internally emulated machines to T_2 .
- T_2 forwards messages from its internally emulated machines to T_1 , subject to the (per-entity) bounds of π according to Definition 26.

Moreover,

- T_2 passes its input $(1^\kappa, z)$ as input to its internally emulated environment and
- eventually outputs what the internally emulated environment outputs.

Then, the interaction between T_1 and T_2 has $O(k)$ rounds according to Definition 16.

Proof. By definition, the communication between T_1 and T_2 consists of the following messages:

- The initial message from T_1 to T_2 .
- Inputs and outputs of (honest) main parties of π .
- Messages reported by the dummy adversary to the environment related to π .
- Messages sent from the environment to the dummy adversary related to π .

Using Definition 25 and the fact that π is a k -round protocol, it is easy to see that the number of these messages is bounded by $O(k)$.

We now split up \mathcal{E}'_1 like in Proposition 7. To this end, let \mathcal{E}'_2 be an ITM that is identical to \mathcal{E}'_1 , except with the following differences:

- Externally, interact with a protocol and an adversary.
- Internally, execute an adversary \mathcal{D}' that is defined as follows:
 - Messages from the environment to corrupted parties intended for \mathcal{H} (and vice versa) are forwarded between \mathcal{H} and \mathcal{Z} .
- Forward messages from the internally emulated machines to an external machine T_1 , subject to the (per-entity) bounds of π according to Definition 26.

Here, \mathcal{O}_{CCA} does not have access to the views of the (external) protocol and adversary. We can see the external protocol and (dummy) adversary resp. simulator (for the dummy adversary) as machine T_1 and \mathcal{E}'_2 as T_2 in Proposition 7 and conclude that the interaction between external protocol and (dummy) adversary on the left side and \mathcal{E}'_2 has $O(k)$ rounds. Also, it is easy to see that \mathcal{E}'_2 is PPT. By using the black-box property and the $O(k)$ -robust composition-order invariance of \mathcal{O}_{CCA} , it thus follows that the statistical distance between the output of \mathcal{E}'_1 and \mathcal{E}'_2 is negligible.

By using the $O(k)$ -robust pseudo-PPT property of \mathcal{O}_{CCA} (Definition 6), we can replace \mathcal{E}'_2 with access to \mathcal{O}_{CCA} with a PPT ITM \mathcal{E}'_3 without access to any (pseudo-)oracle, incurring a negligible change in the statistical distance between the outputs only.

Let \mathcal{Z}' be the UC environment that internally executes \mathcal{E}'_3 and relays messages between \mathcal{E}'_3 and the challenge protocol and adversary appropriately. Eventually, \mathcal{Z}' runs the (unbounded) D on the output of \mathcal{E}'_3 and outputs what D outputs. We obtain a distinguishing UC environment \mathcal{Z}' from an Rewinding UC environment \mathcal{Z} and distinguisher D , leading to a contradiction of the fact that π statistically UC-emulates ϕ .

Of course, compatibility is not limited to the cases mentioned in Theorem 7 and its variants. However, manual proofs may be necessary.

Meaningfulness. Just like the angel in [PS04] or the helper in [CLP10], our helper may negatively affect the security guarantees provided by ideal functionalities. To illustrate this, consider a variant \mathcal{F}'_{COM} of the ideal functionality for commitments \mathcal{F}_{COM} , which we extend to accept a CRS from the adversary. When the honest

committer provides its input v , $\mathcal{F}'_{\text{COM}}$ first checks if the CRS is a valid CRS for the statistically hiding commitment scheme COM of \mathcal{H}^{24} . Then, it performs the commit phase with the adversary, acting as an honest committer with input v .

In the presence of \mathcal{H} , $\mathcal{F}'_{\text{COM}}$ provides no meaningful security. The adversary simply can start a new session with the committed-value oracle provided by \mathcal{H} , receiving a valid CRS which it provides to $\mathcal{F}'_{\text{COM}}$. Then, it can forward all commitment-related messages between \mathcal{H} and $\mathcal{F}'_{\text{COM}}$. In the end, the adversary will learn v , i.e. the value committed to by the honest committer, from \mathcal{H} . (The argument for [CLP10; PS04] is analogous.)

Thus, (long-term) Rewinding UC security only guarantees meaningful security for ideal functionalities with less than or equal to k rounds if \mathcal{O}_{CCA} (in \mathcal{H}) is $O(k)$ -robust, pseudo-PPT $O(k)$ -robust composition-order invariant. Note that very similar limitations with respect to the meaningfulness apply to e.g. [CLP10; PS04].

Justification. We now discuss under which circumstances our notion implies existing security notions for (composable) multi-party computation. This is helpful to grasp the (intuitive) security guarantees of (long-term) Rewinding UC security. First, we show that Rewinding UC security implies UC security for a large class of protocols.

Proposition 8 (Justification: UC Security). *Let π, ϕ be PPT protocols such that $\pi \geq_{\text{R}} \phi$ (resp. $\pi \geq_{\text{R}}^{\text{lt}} \phi$) and the simulator never needs to interact with \mathcal{H} on the committed-value oracle for the challenge session. Then, $\pi \geq_{\text{UC}} \phi$ (resp. $\pi \geq_{\text{ltUC}} \phi$).*

Proof. We prove Proposition 8 only for the standard non-long-term notion. The proof for the other case is similar.

Let π, ϕ be protocols such that $\pi \geq_{\text{R}} \phi$ and the simulator \mathcal{S} (for the dummy adversary) never queries \mathcal{H} on the committed-value oracle for the challenge session. Suppose that for the sake of contradiction it holds that $\pi \not\geq_{\text{UC}} \phi$, i.e. for all (presumptive) PPT UC simulators \mathcal{S}' for the dummy adversary, there exists an environment \mathcal{Z} that can distinguish between the UC execution of π and \mathcal{D} and the UC execution of ϕ and \mathcal{S}' .

We construct an environment \mathcal{Z}' that distinguishes between the Rewinding UC execution of π and \mathcal{D} and the Rewinding UC execution of ϕ and \mathcal{S} as follows:

- On input $(1^\kappa, z)$, activate \mathcal{Z} on input $(1^\kappa, z)$.
- Whenever \mathcal{Z} corrupts a party, send an appropriate `corrupt` message to \mathcal{H} .
- Relay all messages between \mathcal{Z} , the challenge protocol and the adversary.
- Output whatever \mathcal{Z} outputs.

²⁴ Here, we assume that a CRS that leads to a statistically hiding commitment scheme is efficiently recognizable.

As \mathcal{Z} is a UC environment, it never queries \mathcal{H} or instructs the dummy adversary to do so. By assumption, neither does \mathcal{S} query the committed-value (pseudo-)oracle of \mathcal{H} . Thus, in the execution with \mathcal{Z}' , \mathcal{S} behaves like an UC simulator and the view of \mathcal{Z} is correctly distributed as in an UC execution with the challenge protocol and the dummy adversary resp. the (presumptive) simulator with the dummy adversary. As a consequence, the distinguishing advantage of \mathcal{Z}' in the Rewinding UC execution is identical to the distinguishing advantage of \mathcal{Z} , leading to a contradiction.

For the case of ideal functionalities that can be expressed by stand-alone real-ideal security (see e.g. [Gol04]), the following holds regardless of the simulator using the committed-value oracle of \mathcal{H} .

Proposition 9 (Justification: Stand-Alone Security for SFE). *Let \mathcal{H} be a helper with a committed-value oracle that is black-box and $O(1)$ -robust composition-order invariant and pseudo-PPT. Let π be a N -party PPT protocol in the \mathcal{F}_{CRS} -hybrid model such that π (long-term-) Rewinding-UC-realizes \mathcal{F}_{SFE} (with \mathcal{H}) for some function $f : (\{0, 1\}^\kappa)^N \times \{0, 1\}^{\text{poly}(\kappa)} \rightarrow (\{0, 1\}^\kappa)^N$. Then, π securely computes f with abort in the presence of static malicious adversaries.*

In particular, Proposition 9 captures the stand-alone real-ideal security of e.g. zero-knowledge proof systems. The restriction to protocols in the \mathcal{F}_{CRS} -hybrid model can be relaxed to other hybrid functionalities that can be expressed by stand-alone real-ideal security.

We omit the proof of Proposition 9, but note that that the distinguisher in the real-ideal security notion is not provided with a committed-value oracle (corresponding to an Rewinding UC environment that never queries the committed-value oracle of \mathcal{H}). Thus, the (PPT) simulator may only need to extract commitments for its own simulation, which it can do efficiently via rewinding, regardless of the number of rounds of π .

Environmental Friendliness. Similar to [CLP10], our notion partially fulfills the notion of environmental friendliness [CLP13]. Suppose that the committed-value oracle of \mathcal{H} is $O(k)$ -robust, pseudo-PPT $O(k)$ -robust composition-order invariant and that a PPT protocol π (long-term-) Rewinding-UC-realizes an ideal functionality \mathcal{G} . Then, we can show that for every k -round game-based property of a protocol that is executed concurrently (outside the Rewinding UC execution), the protocol π does not affect this game-based property if it is not already affected by \mathcal{G} (in an execution *without* \mathcal{H}). For details, see Appendix D.3.

The following is largely based on the full version of [BMM21]. First, we restate the notion of a *security game* as defined in [CLP13].

Definition 27 (Security Game, [CLP13]). *A security game (or game) consists of an ITM Chal , called the challenger, that is polynomial-time in the length of the messages it receives, and a constant $\tau_{\mathcal{C}}$, called the threshold, in the interval $[0, 1)$. In an execution of a security game, the challenger Chal*

interacts with an adversary A on common input 1^n and outputs accept or reject at the end of the interaction.

We say that A_n breaks Chal_n with advantage ε , if A_n makes Chal_n accept with probability $\tau_C + \varepsilon$. We say that A breaks Chal , or the game-based assumption \mathcal{C} , if A_n breaks Chal_n with advantage $\varepsilon(n)$ for infinitely many $n \in \mathbb{N}$ for a non-negligible function ε . ε is the advantage of the adversary.

An example for such a security game could be the IND-CPA game for encryption schemes with $\tau = 1/2$, i.e. the trivial winning probability of an adversary. However, Definition 27 is also valid for IND-CPA security with $\tau = 1$.

Based on *games*, one can define *assumptions*, which restrict the parameter τ such that there exists a trivial strategy for adversaries to win the game with probability τ .

Definition 28 (Game-Based Assumptions, [CLP13]). A game-based assumption is simply a security game $\mathcal{C} = (\text{Chal}, \tau)$, such that, there is a non-uniform PPT adversary A , called the trivial strategy, satisfying that A_n breaks Chal_n with probability at least τ (possibly without any advantage) for all $n \in \mathbb{N}$. We say that assumption \mathcal{C} holds if no non-uniform PPT adversary can break the game (Chal, τ) .

Definition 28 would rule out $\tau = 1$ for IND-CPA security, as there is no trivial winning strategy for the IND-CPA game with winning probability 1. However, the definition of game-based assumptions does not rule out the existence of insecure schemes Π for which the adversary has a non-negligible advantage over τ . This is covered in the following definition of *game-based security properties*.

Definition 29 (Game-Based Security Property, [CLP13]). A game-based security property of a cryptographic scheme Π is simply a security game $P_\Pi = (\text{Chal}, \tau)$. We say that the property P_Π holds if no non-uniform PPT adversary can break the game (Chal, τ) .

However, the game for IND-CPA security does not capture a setting where other protocols are executed concurrently. In order to argue that the security of Π is not impacted by a protocol ρ that runs concurrently and implements a functionality \mathcal{G} , the game P_Π has to be modified accordingly. The proceedings version [CLP13] gives an informal description of the associated game (see the full version²⁶ for a complete description):

Similar to UC security, an environment Z that gives input to the challenger Chal as well as ρ and may freely interact with the adversary A , is introduced. The adversary A not only interacts with Chal , but also with ρ (which may be a “real” protocol or the ideal protocol of some functionality \mathcal{G}). Z may, in particular, correlate the inputs of ρ and Chal . However, Π and ρ are never sub-routines of one another. As a consequence, environmental

friendliness does not generally imply composability in the sense of subroutine replacement. Also, the adversary A does not “attack” the execution of ρ as in the UC execution experiment. Furthermore, there is no simulator. Like in the security game (Chal, τ) , the adversary’s success is determined by the output of Chal and not e.g. by the output of Z .

The game $\text{Chal}^{\mathcal{G}/\rho}$ is defined similar to Chal , with the exception that (the ideal protocol of) \mathcal{G} is replaced with ρ . In contrast to UC security, the environment Z and the adversary know whether \mathcal{G} or ρ are executed.

The outlined game is (implicitly) considered in the following definition.

Definition 30 (Environmental Friendliness, [CLP13]). *Let $P = (\text{Chal}, \tau)$ be a game-based security property of a cryptographic scheme Π , and ρ a protocol implementing a functionality \mathcal{G} . Then we say that ρ is environmental friendly to Π with property P , if the security property $P^{\mathcal{G}/\rho} = (\text{Chal}^{\mathcal{G}/\rho}, \tau)$ holds.*

Proposition 10 (Environmental Friendliness of (long-term) Rewinding UC Security). *Let \mathcal{H} be a helper where the committed-value (pseudo-)oracle provided by \mathcal{H} is black-box (Definition 4), $O(k)$ -robust composition-order invariant (Definition 5) and pseudo-PPT (Definition 6). Let π be a protocol that (long-term-) Rewinding-UC-emulates the ideal protocol of some functionality \mathcal{G} (with respect to \mathcal{H}). Then π is friendly to every k -round game-based property P of a protocol Π with property P .*

The intuition behind Proposition 10 is as follows. Suppose that a game-based property P holds in the execution with \mathcal{G} , but not in the execution with π . We can then use the Rewinding UC simulator and \mathcal{G} to emulate π (with the help of \mathcal{H}), incurring at most a negligible difference in the adversary’s success. As a next step, we use the robustness of \mathcal{O}_{CCA} within \mathcal{H} to replace the simulator \mathcal{S} with access to \mathcal{H} with an efficient simulator²⁶ \mathcal{S}' . This again incurs only a negligible difference in the adversary’s success. We again arrive at an execution with \mathcal{G} and a PPT adversary, leading to a contradiction because P holds in an execution with \mathcal{G} by assumption.

The proof of Proposition 10 is very similar to the proof of [CLP13, Theorem 7] (in the full version) and the proof of Theorem 7 and we thus omit it.

Impossibility Results. While the addition of the helper \mathcal{H} , which allows the extraction of statistically hiding commitments, suffices to “circumvent” the impossibility results of Müller-Quade and Unruh [MU10], our setting still faces an important impossibility result for *long-term* Rewinding UC.

²⁶ <https://www.cs.cornell.edu/~rafael/papers/EnvFriendly-proc.pdf>

²⁶ For this argument, we only need the committed-value oracle of \mathcal{H} , but not its complexity oracles

Theorem 8. *Let \mathcal{F} be a functionality that is long-term revealing (Definition 21) for any party. Then, there is no bilateral²⁷ nontrivial PPT protocol π_{OT} that long-term-Rewinding-UC-realizes \mathcal{F}_{OT} in the \mathcal{F} -hybrid model (assuming ideally authenticated communication).*

Theorem 8 is a direct consequence of the folklore impossibility result of correct statistically secure oblivious transfer in the plain model (even with passive security only).

In the following, we give a formal proof, using a similar approach to the one in [MU10]. We note that we can extend Theorem 8 to the case of ideally secure communication using a slightly different proof (where the adversary passively corrupts parties to obtain the communication).

Proof. For a protocol π_{OT} to long-term-Rewinding-UC-realize \mathcal{F}_{OT} , π_{OT} must simultaneously fulfill the properties of i) correctness, ii) long-term sender security and iii) long-term receiver security. We show that these properties cannot be fulfilled simultaneously if \mathcal{F} is long-term-revealing for either party.

To this end, we consider an execution of π_{OT} with an environment \mathcal{Z} and the dummy adversary on security parameter κ where \mathcal{Z} (i) instructs the dummy adversary to immediately deliver all messages, (ii) never instructs the dummy adversary to corrupt a party and (iii) never interacts with \mathcal{H} (i.e. does not extract commitments), (iv) receives (external) input (m_0, m_1, b) and uses (m_0, m_1) as input for the (honest) OT sender and b as input for the (honest) OT receiver. As usual, all communication between parties goes either through the adversary or through \mathcal{F} .

We use the following notation, based on [MU10, Section 4.1]:

- $\text{COM}^{m_0, m_1, b}(\dots)$ denotes the communication of the parameterized machine pairs in the above execution when the OT input of the receiver is b and the input of the sender is (m_0, m_1) . For example, $\text{COM}^{m_0, m_1, b}(\mathcal{Z} \text{S}, \text{S} \mathcal{A}, \text{S} \mathcal{F})$ contains all communication of the sender, which consists of the communication between \mathcal{Z} and S (inputs and subroutine outputs), between S and \mathcal{A} (messages) and S and \mathcal{F} (inputs and subroutine outputs).
- $\text{OUT}^{m_0, m_1, b}$ denotes the output of the receiver.
- For families of variables $A_{\kappa, z}$ and $B_{\kappa, z}$, we write $A \triangleright B$ if there is some probabilistic function G such that $B_{\kappa, z} \stackrel{\text{S}}{\approx} G(\kappa, A_{\kappa, z})$ (and vice versa for \triangleleft). It is easy to see that \triangleright and \triangleleft are transitive. For the sake of an easier notation, we will ignore κ from now on.

We first establish several properties.

²⁷ We recall the definition of a bilateral protocol due to Canetti and Fischlin [CF01]: “[A] protocol π between n parties P_1, \dots, P_n is *bilateral* if all except two parties stay idle and do not transmit messages.”

For an OT protocol with long-term receiver security, it holds for all m_0, m_1 that

$$\text{COM}^{m_0, m_1, 0}(\mathcal{ZS}, \mathcal{SA}, \mathcal{SF}) \stackrel{s}{\approx} \text{COM}^{m_0, m_1, 1}(\mathcal{ZS}, \mathcal{SA}, \mathcal{SF}) \quad (1)$$

Conversely, for long-term sender security, it holds for all m_b, m_{1-b}, m'_{1-b} and $b \in \{0, 1\}$ that²⁸

$$\text{COM}^{m_0, m_1, b}(\mathcal{ZR}, \mathcal{RA}, \mathcal{RF}) \stackrel{s}{\approx} \text{COM}^{m_b, m'_{1-b}, b}(\mathcal{ZR}, \mathcal{RA}, \mathcal{RF}) \quad (2)$$

For a correct protocol, it must hold that for all m_0, m_1 and all $b \in \{0, 1\}$ that

$$m_b \stackrel{s}{\approx} \text{OUT}^{m_0, m_1, b} \triangleleft \text{COM}^{m_0, m_1, b}(\mathcal{RA}, \mathcal{RF}) \quad (3)$$

$$\text{COM}^{m_0, m_1, b}(\mathcal{SA}, \mathcal{SF}) \triangleright m_b \quad (4)$$

and, if $m_0 \neq m_1$,

$$m_0 \stackrel{s}{\approx} \text{OUT}^{m_0, m_1, 0} \not\stackrel{s}{\approx} \text{OUT}^{m_0, m_1, 1} \stackrel{s}{\approx} m_1 \quad (5)$$

where Eq. (3) means that the receiver's output (but not necessarily b) can be reconstructed with overwhelming probability from the receiver's communication, Eq. (4) means that (at least) the result m_b can be reconstructed from the sender's communication (including its communication with \mathcal{F}) and Eq. (5) guarantees that if m_0 and m_1 differ, then, for different choice bits, the output of the receiver will be (statistically) different.

Claim 4. If \mathcal{F} is long-term-revealing for R and π_{OT} is long-term receiver-secure, then π_{OT} cannot be correct.

Proof. If \mathcal{F} is long-term-revealing for R, it holds that

$$\text{COM}^{m_0, m_1, b}(\mathcal{RA}, \mathcal{RF}) \triangleleft \text{COM}^{m_0, m_1, b}(\mathcal{RA}, \mathcal{SF}) \quad (6)$$

i.e. the communication between R and \mathcal{F} can be computed from the communication between S and \mathcal{F} . As the communication between R and \mathcal{A} can be computed from the communication between S and \mathcal{A} , it holds that

$$\text{COM}^{m_0, m_1, b}(\mathcal{RA}, \mathcal{SF}) \triangleleft \text{COM}^{m_0, m_1, b}(\mathcal{SA}, \mathcal{SF}) \quad (7)$$

Combining Eqs. (1), (3), (6) and (7), using the definition of \triangleleft and the transitivity of indistinguishability, we obtain

$$\begin{aligned} m_0 &\stackrel{s}{\approx} \text{OUT}^{m_0, m_1, 0} \stackrel{s}{\approx} G(\text{COM}^{m_0, m_1, 0}(\mathcal{SA}, \mathcal{SF})) \\ &\stackrel{s}{\approx} G(\text{COM}^{m_0, m_1, 1}(\mathcal{SA}, \mathcal{SF})) \stackrel{s}{\approx} \text{OUT}^{m_0, m_1, 1} \\ &\stackrel{s}{\approx} m_1 \end{aligned} \quad (8)$$

which contradicts Eq. (5), i.e. the correctness.

Claim 5. If \mathcal{F} is long-term revealing for \mathcal{S} and π_{OT} is correct and long-term receiver-secure, then π_{OT} cannot be long-term sender-secure.

Proof. Using that \mathcal{F} is long-term revealing for \mathcal{S} and that the communication between \mathcal{S} and \mathcal{A} can be computed from the communication between \mathcal{R} and \mathcal{A} , it follows that for $b \in \{0, 1\}$

$$\text{COM}^{m_0, m_1, b}(\mathcal{R}\mathcal{A}, \mathcal{R}\mathcal{F}) \triangleright \text{COM}^{m_0, m_1, b}(\mathcal{S}\mathcal{A}, \mathcal{S}\mathcal{F}) \triangleright m_b \quad (9)$$

Combining Eqs. (1), (4) and (9) and using the definition of \triangleright and the transitivity of indistinguishability, it follows that

$$m_0 \stackrel{s}{\approx} G(\text{COM}^{m_0, m_1, 1}(\mathcal{R}\mathcal{A}, \mathcal{R}\mathcal{F})) \quad (10)$$

which contradicts Eq. (2), i.e. the sender security (because m_0 can be computed from \mathcal{R} 's interaction with \mathcal{A} and \mathcal{F} , even though its choice bit was 1).

Combining Claims 4 and 5, the theorem follows.

Remark 23. While we have considered the case of long-term security, the proof similarly holds for statistical security. As all parties are honest and by considering an appropriate environment, there is no communication with \mathcal{H} and it can thus be ignored.

E Proof of Theorem 5

Proof. In the following, we prove Theorem 5.

We assume static corruptions and can thus distinguish between the corrupted parties in the following proof. We obtain a simulator \mathcal{S} for all possible corruptions by combining the individual simulators. As the dummy adversary is complete (see Proposition 5), we consider simulators for the dummy adversary.

Corrupted Committer. We now state the simulator for the dummy adversary and a corrupted committer.

Definition 31 (Simulator for the Dummy Adversary, Corrupted Committer, Honest Receiver).

1. Handle messages between \mathcal{Z} and \mathcal{H} like the dummy adversary.
2. Report all messages coming from internally simulated honest parties to the environment and wait for its confirmation to deliver them. Until the reported message of the honest party is delivered, pause the simulation of this party.

²⁸ In abuse of notation, we write $\text{COM}^{m_b, m'_{1-b}, b}$ to denote $\text{COM}^{m_0, m'_1, 0}$ resp. $\text{COM}^{m'_0, m_1, 1}$.

3. Deliver messages as instructed by the environment to internal simulations of the honest party.
4. Send $(\mathbf{ext-init}, \mathbf{C}, \mathit{sid}, r \stackrel{\$}{\leftarrow} \{0,1\}^\kappa)$ to \mathcal{H} in the name of \mathbf{C} and receive $(\mathbf{setup}, \mathit{sid}, \mathit{ck})$ from \mathcal{H} . Report ck as output of \mathcal{F}_{CRS} with $\text{SID } \mathit{sid} \parallel \mathbf{crs}$. (When the environment has already queried \mathcal{H} on sub-session $(1, r)$, use a different r' instead. If the environment later queries \mathcal{H} on sub-session $(1, r)$, use a different r' and report the answers with r instead of r' .)
5. Commit phase: Let m denote a commitment message received from the corrupted committer and send $(\mathbf{ext-mesg}, \mathbf{C}, \mathit{sid}, m)$ to \mathcal{H} . If \mathcal{H} answers with $(\mathbf{ext-mesg}, \mathbf{C}, \mathit{sid}, m')$, report m' as message from \mathbf{R} to \mathbf{C} . If \mathcal{H} answers with $(\mathbf{ext-val}, \mathit{sid}, 1, v, \mathit{view})$,
 - output a special symbol \perp if $v = \perp_{\text{ext}}$,
 - halt the simulation of the receiver if $v' = \perp$, i.e. the receiver would not accept the commitment,
 - Otherwise, send $(\mathbf{commit}, \mathit{sid}, v)$ to \mathcal{F}_{COM} on behalf of \mathbf{C} . Also allow the committed output of \mathcal{F}_{COM} for the receiver.
6. Eventually receive a message $(\mathbf{unveil}, \mathit{sid}, v', d')$ from the committer and proceed as follows:
 - If $v' = v$ and the honest receiver would accept, send $(\mathbf{unveil}, \mathit{sid})$ to \mathcal{F}_{COM} and allow the output.
 - If $v' \neq v$ and the honest receiver would accept, output a special error symbol \perp .

In order to prove the validity of the simulator \mathcal{S} in Definition 31, we define a number of hybrids. We start with the real execution of π_{COM} and the dummy adversary \mathcal{D} and gradually change it to an execution of \mathcal{F}_{COM} and the simulator \mathcal{S} . For each pair of hybrids, we prove the statistical indistinguishability.

- H_0 : The real execution with π_{COM} and \mathcal{D} .
- H_1 : Execution with the ideal functionality \mathcal{F}_1 that lets the adversary determine all inputs and learn all outputs. \mathcal{S}_1 is the simulator that executes the protocol π_{COM} honestly on behalf of the honest party, using the inputs learned from \mathcal{F}_1 and making the outputs through \mathcal{F}_1 . Messages related to \mathcal{H} are handled like by the dummy adversary.
- H_2 : The ideal execution with \mathcal{F}_{COM} and \mathcal{S} .

Claim 6. If \mathcal{O}_{CCA} is black-box, then out_0 and out_1 are identically distributed.

Proof. As the changes between H_0 and H_1 are only syntactic and oblivious for the environment, the claim follows due to the black-box property of \mathcal{O}_{CCA} .

Claim 7. Let \mathcal{O}_{CCA} be a black-box committed-value pseudo-oracle for COM . If COM is a CCA-binding commitment scheme (Definition 7) with respect to \mathcal{O}_{CCA} , then $\text{out}_1 \stackrel{\$}{\approx} \text{out}_2$.

Proof. It is easy to see from the definition of \mathcal{S} and the black-box property of \mathcal{O}_{CCA} that out_1 and out_2 are identically distributed unless \mathcal{S} outputs \perp in H_2 . Let \mathbf{E}_\perp denote this event.

We show that $\Pr[E_{\perp}] \leq \text{negl}$ for some negligible function negl . To this end, we construct an adversary \mathcal{B} against the CCA binding property that includes the execution H_1 , but plays the commitment it receives from the corrupted committer with the experiment. After the commit phase has finished, it receives either the extracted value $e \in \{0, 1\}^{\kappa}$, a special error symbol \perp_{ext} (if the commitment could not be extracted) or \perp if the receiver did not accept. We distinguish between the following cases for the extracted value:

- $e = \perp$: The receiver would not accept and the execution would not continue. Thus, also halt.
- $e = \perp_{ext}$: If the commitment gets unveiled later on, send the unveil information to the game, winning it.

By definition, \mathcal{B} wins the CCA-binding game if E_{\perp} occurs. Thus, $\Pr[E_{\perp}]$ can be bounded by the success probability of an adversary in the CCA-binding game.

We now give a formal proof based on the above intuition.

Let \mathcal{B} be the following adversary against the CCA-binding property of COM.

1. Initially, set $e = \perp$.
2. On input $(1^{\kappa}, z)$, emulate an execution of H_1 with input $(1^{\kappa}, z)$ for the environment. In deviation from H_2 , perform all commitments with a corrupted committer (i.e. the commitment of the challenge session as well as commitments with \mathcal{H}) with \mathcal{O}_{CCA} . We only sketch how the challenge session is handled. To this end, start a session with \mathcal{O}_{CCA} . Initially, receive a CRS ck from \mathcal{O}_{CCA} . Report ck_i as output of \mathcal{F}_{CRS} with $SID \text{ } sid || crs$ where sid is the SID of the challenge session in H_1 . At the end of the commit phase, set e to the extracted value returned by \mathcal{O}_{CCA} and let c denote the corresponding transcript.
3. Simulate \mathcal{H} has follows:
 - Appropriately forward `ext-init` and `ext-mesg` messages between \mathcal{O}_{CCA} and \mathcal{Z} , exposing the same interface to \mathcal{Z} as with \mathcal{H} .
4. After the commit phase has finished, do the following:
 - If $e = \perp$, i.e. the receiver would not accept, halt.
 - If $e = \perp_{ext}$ or $e = v \in \mathcal{M}$, continue.
5. If the corrupted committer eventually performs the unveil phase by sending (v', d') , send (v', d') to the CCA binding game as unveil message.

It is easy to see that the view of the internally simulated environment \mathcal{Z} is distributed as in H_1 . By the definition of \mathcal{B} and the black-box property of \mathcal{O}_{CCA} ²⁹, it thus holds that its advantage in the CCA binding game is greater than or equal to $\Pr[E_{\perp}]$. As COM is CCA binding by assumption, we can thus bound $\Pr[E_{\perp}]$ by a negligible function $\text{negl}_{CCA\text{-binding}}$ for the advantage of an adversary in the CCA-binding game. The claim follows.

²⁹ Formally, we cannot apply the black-box property as-is. This is due to the fact that the number of queries to \mathcal{O}_{CCA} changes between hybrids H_1 and H_2 . This problem can be solved by introducing an intermediate hybrid where all commitments with the corrupted committer are forwarded to \mathcal{O}_{CCA} , but the extracted value discarded. Clearly, this does not change the distribution. Coming from this hybrid, we can apply the black-box property.

As the number of hybrids is constant, it follows that $\text{out}_0 \stackrel{s}{\approx} \text{out}_2$ in case of a corrupted committer.

Corrupted Receiver. We now state the simulator for the dummy adversary and a corrupted receiver.

Definition 32 (Simulator for the Dummy Adversary, Corrupted Receiver).

- Handle messages between \mathcal{Z} and \mathcal{H} like the dummy adversary.
- Report all messages coming from internally simulated honest parties to the environment and wait for its confirmation to deliver them. Until the reported message of the honest party is delivered, pause the simulation of this party.
- Deliver messages as instructed by the environment to internal simulations of the honest party.
- Initially, sample $(\text{ck}, \text{td}) \leftarrow \text{TSetup}(1^\kappa)$. Report ck as output of \mathcal{F}_{CRS} with $\text{SID } \text{sid} \parallel \text{crs}$.
- When receiving the message $(\text{committed}, \text{sid})$ from \mathcal{F}_{COM} , use the trapdoor committer algorithm C_{trap} of COM on input $(1^\kappa, \text{ck}, \kappa, \text{td})$ to perform the commitment with the corrupted receiver. After the commit phase has finished and all messages have been delivered, allow the output.³⁰
- When receiving the message $(\text{unveil}, \text{sid}, v)$ from \mathcal{F}_{COM} : Use the trapdoor committer C_{trap} of COM to create unveil information d for a commitment to v and send (v, d) to the receiver.

To show that the simulator is valid, we consider the following hybrids and prove their statistical indistinguishability:

- H_0 : The real execution with π_{COM} and \mathcal{D} .
- H_1 : Execution with the ideal functionality \mathcal{F}_1 that lets the adversary determine all inputs and learn all outputs. \mathcal{S}_1 is the simulator that executes the protocol π_{COM} honestly on behalf of the honest party, using the inputs learned from \mathcal{F}_1 and making the outputs through \mathcal{F}_1 . Messages related to \mathcal{H} are handled as by the dummy adversary.
- H_2 : \mathcal{F}_2 is identical to \mathcal{F}_1 . \mathcal{S}_2 is defined as \mathcal{S}_1 , but uses the algorithm of the trapdoor committer of COM like the simulator in Definition 32.
- H_3 : The ideal execution with \mathcal{F}_{COM} and \mathcal{S} .

Claim 8. If \mathcal{O}_{CCA} is black-box, then out_0 and out_1 are identically distributed.

Proof. As the changes between H_0 and H_1 are only syntactic and oblivious for the environment, the claim follows.

Claim 9. Let \mathcal{O}_{CCA} be a black-box committed-value pseudo-oracle for COM.

If COM is a trapdoor commitment scheme with respect to \mathcal{O}_{CCA} , then out_1 and out_2 are statistically indistinguishable.

³⁰ As R is corrupted, allowing the output has no visible effect. However, it is necessary for \mathcal{F}_{COM} to continue.

Proof. We prove Claim 9 by a reduction to the trapdoor property of COM (Definition 8) with respect to the pseudo-oracle \mathcal{O}_{CCA} .

Let $(\mathcal{B}^{\mathcal{O}_{\text{CCA}}})$ be the following PPT adversary against the trapdoor property (Definition 8) of the commitment scheme COM:

1. Handle messages for \mathcal{H} like the dummy adversary. In order to emulate \mathcal{H} , use the pseudo-oracle \mathcal{O}_{CCA} provided by the game.
2. On input $(1^\kappa, z)$, internally start an execution of H_2 , but perform the commitment with the game. More specifically, send (**Setup**) to obtain a setup ck and report ck as output of the instance of \mathcal{F}_{CRS} with SID $\text{sid}||n||\text{crs}$.
3. At the beginning of the commit phase, receive the honest committer's input v and send (**Start**, v) as challenge to the game and forward messages between the game and the corrupted receiver.
4. When receiving the message (**unveil**, sid, v) from \mathcal{F}_2 , send (**Unveil**) to the game and obtain the unveil information d . Send (v, d) to the corrupted receiver.
5. Continue the execution.

If the challenge bit b in the TDC game is 0, then the view of the internally emulated environment is distributed as in an execution of H_1 due to the black-boxness of \mathcal{O}_{CCA} . If $b = 1$, it is distributed in H_2 . It follows from a standard argument that if $\text{out}_1(\kappa, z) \not\stackrel{s}{\approx} \text{out}_2(\kappa, z)$, then $\text{out}_0^{\text{TDC}}(\kappa, z) \not\stackrel{s}{\approx} \text{out}_1^{\text{TDC}}(\kappa, z)$, i.e. the output of the trapdoor game with choice bit b and input (κ, z) , contradicting the trapdoor property of COM.

Proof. As the changes between H_2 and H_3 are only syntactic and oblivious for the environment, the claim follows.

As there is a common bound between the hybrids, it follows that $\text{out}_0 \stackrel{s}{\approx} \text{out}_3$ and the claim follows.

Both parties honest. This case is very similar to the case of the corrupted receiver and we omit it.

F Applications

We present several applications of our long-term composable commitment scheme.

F.1 Zero-Knowledge and Commit-and-Prove

By plugging in our commitment scheme into an appropriate zero-knowledge proof system in the \mathcal{F}_{COM} -hybrid model with statistical UC security, e.g. the construction of Canetti and Fischlin [CF01], we obtain the following theorem.

Theorem 9. *Assume that computationally binding, statistically hiding trapdoor commitment schemes with public-coin receiver and non-interactive unveil phase exist (in the \mathcal{F}_{CRS} -hybrid model). Then, for every NP relation R , there exists a protocol π'_R in the \mathcal{F}_{CRS} -hybrid model such that π'_R long-term-Rewinding-UC-realizes $\mathcal{F}_{\text{ZK}}^R$.*

The resulting protocol π'_R thus features statistical zero-knowledge and knowledge soundness against computationally bounded provers.

Using a similar approach, we obtain a protocol that long-term-realizes the ideal functionality for commit-and-prove (Definition 15) for a bounded number of proofs per instance.

Theorem 10. *Let us assume that computationally binding, statistically hiding trapdoor commitment schemes with public-coin receiver and non-interactive unveil phase exist (in the \mathcal{F}_{CRS} -hybrid model). Let $k \in \text{poly}(\kappa)$. There exists a protocol SCP' (accepting at most k inputs) in the \mathcal{F}_{CRS} -hybrid model such that SCP' long-term-Rewinding-UC-realizes \mathcal{F}_{CP} (for up to k inputs).*

F.2 Oblivious Transfer with Long-term Security for One Party

In the following, we construct a protocol for composable oblivious transfer where one party is protected with long-term security, which is the best we can hope for due to the impossibility result of our notion (Theorem 4).

Oblivious Transfer with Long-Term Security for one Party

Even given a commitment scheme that is long-term-Rewinding-UC-secure, we cannot hope to achieve long-term-secure secure function evaluation from long-term-revealing setups when at least one party's output depends on inputs from both parties.

However, we can construct an OT protocol π_{tOT} that at least guarantees long-term security for one party. Before presenting the protocol, we first discuss the difficulties of defining security in this setting.

Defining Security. In the case of long-term security, an unbounded distinguisher is given (w.l.o.g.) the environment's view, which may contain messages from honest parties that are only given computational privacy. For long-term emulation of the ideal functionality to hold, these messages must be statistically indistinguishable in the real resp. ideal execution. Thus, the long-term simulation must statistically depend on the honest parties' secrets, which are usually not available to the simulator in an execution with an ideal functionality.

As a consequence, we cannot realize e.g. \mathcal{F}_{OT} or \mathcal{F}_{SFE} in our framework with long-term security using long-term-revealing setups only. We leave the definition of ideal functionalities with meaningful security guarantees that can be realized in this setting as an interesting future work.

In the following, we will resort to an approach that is closely related to the one proposed by Peikert, Vaikuntanathan, and Waters [PVW08]. In [PVW08], composable oblivious transfer in the \mathcal{F}_{CRS} -hybrid model is constructed. Depending on the CRS distribution, it is possible to either i) achieve universal composability and computational security for both parties or ii) not achieve composability, but statistical security for the OT sender or iii) not achieve composability, but statistical security for the OT receiver.

To prove the security in the two latter cases, the security of the protocol in [PVW08] is proven only for the case that the party with long-term security is honest and the other party (i.e. the one with computational security) is corrupted. The consequences are two-fold: For the corrupted party, the simulator can learn the input in the ideal execution, resulting in a statistically correct simulation. Conversely, it is necessary that the party with statistical security remains honest, because its inputs cannot be extracted in the setting considered by [PVW08]. More concretely, if the OT protocol is, say, statistically sender-secure, a corrupted sender’s input cannot be extracted in a straight-line way, as the sender’s input is statistically hidden. (This is the very observation of Müller-Quade and Unruh [MU10] that protocols in the \mathcal{F}_{CRS} -hybrid model cannot be, *at the same time*, both composable and statistically or long-term-secure when the simulation is straight-line.)

In contrast to [PVW08], we can prove the security of our following constructions in the \mathcal{F}_{CRS} -hybrid model not only in the case that the party with long-term security is honest, but also in the case that this party is (actively) corrupted. This is because we can have *both* protocol parties first commit to their inputs, and, for each step, prove consistency of the following protocol relative to the committed inputs. As our long-term-secure commitment scheme is extractable, we can also learn a corrupted party’s input *without weakening its long-term security*. However, for the reasons outlined above (namely that for the party without long-term security, the simulation must statistically depend on the input), we still assume the party with computational security to be corrupted, too.

When all parties are corrupted, real-ideal security often provides no meaningful security. In order to still prove meaningful guarantees, we make the following assumptions about the ideal execution with \mathcal{F}_{OT} :

1. One party is only passively corrupted.
2. If both parties are corrupted, the simulator only learns the passively corrupted party’s input from the ideal functionality after it has provided the input of the maliciously corrupted party.
3. The output of the passively corrupted party is through the functionality, preventing the simulator from cheating.

Of course, considering both parties to be actively corrupted is possible. However, we cannot hope to express (and achieve) any meaningful security guarantees in such a setting.

This approach gives only a very incomplete overview of the following constructions’ security as we always assume the party with computational security to be (passively) corrupted in order for the simulator to learn its secrets, which provably cannot be protected in the protocol.

Using an appropriate notion, one can show that our constructions indeed provide meaningful security guarantees for parties that are given computational security only. We leave the design of such a notion for future work.

Oblivious Transfer. We start by presenting the protocol π_{tOT} for oblivious transfer with *statistical* UC security for one party in the \mathcal{F}_{CRS} -hybrid model, which works as follows.

First, each party uses an instance of the commit-and-prove functionality \mathcal{F}_{CP} to commit to its input and its randomness. This step enables composability in the first place. Subsequently, both parties execute a protocol π_{OT} in the \mathcal{F}_{CRS} -hybrid model with appropriate properties (e.g. the protocol due to Peikert, Vaikuntanathan, and Waters [PVW08], which provides statistical security for one party but does, in this mode, *not* compose) on their committed inputs and randomness. After each sent message, the message sender proves the correctness relative to the committed input and randomness as well as the previously received messages. The use of \mathcal{F}_{CP} allows for easy extraction resp. equivocation.

By replacing \mathcal{F}_{CP} with an appropriate protocol in the \mathcal{F}_{CRS} -hybrid model, we obtain a composable OT protocol in the \mathcal{F}_{CRS} -hybrid model with long-term Rewinding UC security for one party.

Construction 3 (The OT Protocol π_{tOT})

Parameterized with an actively secure and constant-round OT protocol π_{OT} in the \mathcal{F}_{CRS} -hybrid model.

1. On input (sid, b) , the OT receiver R samples randomness r_R and sends $(\text{commit}, sid || R, (b, r_R))$ to an instance of \mathcal{F}_{CP} with SID $sid || R$.
2. On input $(sid, (m_0, m_1))$, the OT sender S samples randomness r_S and sends $(\text{commit}, sid || S, (m_0, m_1, r_S))$ to an instance of \mathcal{F}_{CP} with SID $sid || S$.
3. R and S keep a list of received messages \overline{m}_R resp. \overline{m}_S .
4. S and R execute π_{OT} on their respective private inputs and randomness r_S resp. r_R .
5. For every message m sent to R by S, S sends $(\text{CP-prover}, sid || S, (m, \overline{m}_S))$ with the relation $R_S^{\pi_{\text{OT}}} = \{(m, \overline{m}_S), (m_0, m_1, r_S) \mid m = \pi_{\text{OT}}((m_0, m_1), \overline{m}_S; r_S)\}$ to prove to R that m is consistent relative to the committed inputs and the received messages. R only continues the execution when it has received $(\text{CP-proof}, sid || S, (m, \overline{m}_S))$.
6. Similarly, for every message m sent to S by R, R sends $(\text{CP-prover}, sid || R, (m, \overline{m}_R))$ with the relation $R_R^{\pi_{\text{OT}}} = \{(m, \overline{m}_R), (b, r_R) \mid m = \pi_{\text{OT}}(b, \overline{m}_R; r_R)\}$ to prove to S that m is consistent relative to the committed inputs and the received messages. S only continues the execution when it has received $(\text{CP-proof}, sid || R, (m, \overline{m}_R))$.
7. When the receiver of π_{OT} outputs m_b , R outputs $(\text{output}, sid, m_b)$.

We first prove that the above construction statistically UC-realizes \mathcal{F}_{OT} if an appropriate protocol π_{OT} is used (and the party without statistical protection in π_{OT} is (passively) corrupted).

Theorem 11. *Let π_{OT}^S be the constant-round maliciously secure OT protocol of [PVW08] with statistical sender security in the \mathcal{F}_{CRS} -hybrid model. Then, π_{tOT} with $\pi_{\text{OT}} = \pi_{\text{OT}}^S$ statistically UC-realizes \mathcal{F}_{OT} for adversaries that at least passively corrupt the receiver R.*

Let π_{OT}^R be the constant-round maliciously secure OT protocol of [PVW08] with statistical receiver security in the \mathcal{F}_{CRS} -hybrid model. Then, π_{tOT} with $\pi_{\text{OT}} = \pi_{\text{OT}}^R$ statistically UC-realizes \mathcal{F}_{OT} for adversaries that at least passively corrupt the sender S.

We do not give a full proof of Theorem 11, but rather the following proof sketch to show the main aspects.

Proof (Sketch). We distinguish between the following cases:

- Protocol π_{ltOT} with $\pi_{\text{OT}} = \pi_{\text{OT}}^{\text{S}}$, i.e. statistical security for the OT sender:
 - Honest sender, passively corrupted receiver: The simulator learns the receiver’s input from \mathcal{F}_{OT} . Otherwise, the simulation is essentially the same as in [PVW08], with the additional handling of \mathcal{F}_{CP} .
 - Honest sender, actively corrupted receiver: The simulation is essentially the same as in [PVW08], with the additional handling of \mathcal{F}_{CP} .
 - Corrupted sender, passively corrupted receiver: The corrupted sender’s input can be extracted from its call to \mathcal{F}_{CP} . As the simulator learns the receiver’s input, it can faithfully execute the real protocol on its behalf. As the corrupted sender is forced to honest behavior due to the use of \mathcal{F}_{CP} and the fact that $\pi_{\text{OT}}^{\text{S}} \geq_{\text{stat-UC}} \mathcal{F}_{\text{OT}}$ for honest senders and (passively) corrupted receivers, real and ideal execution are statistically indistinguishable.
- Protocol π_{ltOT} with $\pi_{\text{OT}} = \pi_{\text{OT}}^{\text{R}}$, i.e. statistical security for the OT receiver:
 - Honest receiver, passively corrupted sender: The simulator learns the sender’s input from \mathcal{F}_{OT} . Otherwise, the simulation is essentially the same as in [PVW08], with the additional handling of \mathcal{F}_{CP} .
 - Honest receiver, actively corrupted sender: The simulation is essentially the same as in [PVW08], with the additional handling of \mathcal{F}_{CP} .
 - Corrupted receiver, passively corrupted sender: The corrupted receiver’s input can be extracted from its call to \mathcal{F}_{CP} . As the simulator learns the sender’s input, it can faithfully execute the real protocol on its behalf. As the corrupted receiver is forced to honest behavior due to the use of \mathcal{F}_{CP} and the fact that $\pi_{\text{OT}}^{\text{R}} \geq_{\text{stat-UC}} \mathcal{F}_{\text{OT}}$ for honest receivers and (passively) corrupted senders, real and ideal execution are statistically indistinguishable.

□

By replacing \mathcal{F}_{CP} with an appropriate long-term-secure protocol, we obtain the following corollaries.

Corollary 3 (Oblivious Transfer with Long-Term Security for the Sender in the \mathcal{F}_{CRS} -hybrid Model). *Let π_{CP} be a protocol that long-term-Rewinding UC-realizes \mathcal{F}_{CP} in the \mathcal{F}_{CRS} -hybrid model. Then, $\pi_{\text{ltOT}}^{\mathcal{F}_{\text{CP}} \rightarrow \pi_{\text{CP}}}$ with $\pi_{\text{OT}} = \pi_{\text{OT}}^{\text{S}}$ Long-Term-Rewinding-UC-realizes \mathcal{F}_{OT} in the \mathcal{F}_{CRS} -hybrid model for adversaries that at least passively corrupt the receiver R.*

Corollary 4 (Oblivious Transfer with Long-Term Security for the Receiver in the \mathcal{F}_{CRS} -hybrid Model). *Let π_{CP} be a protocol that long-term-Rewinding UC-realizes \mathcal{F}_{CP} in the \mathcal{F}_{CRS} -hybrid model. Then, $\pi_{\text{ltOT}}^{\mathcal{F}_{\text{CP}} \rightarrow \pi_{\text{CP}}}$ with $\pi_{\text{OT}} = \pi_{\text{OT}}^{\text{R}}$ Long-Term-Rewinding-UC-realizes \mathcal{F}_{OT} in the \mathcal{F}_{CRS} -hybrid model for adversaries that at least passively corrupt the sender S.*

Observe that π_{tOT} is a constant-round protocol in the $\{\mathcal{F}_{\text{CP}}, \mathcal{F}_{\text{CRS}}\}$ -hybrid model. Using Theorem 7, we import the statistically UC-secure protocol π_{tOT} with an appropriate round complexity into our framework. It follows that $\pi_{\text{tOT}} \geq_{\text{R}}^{\text{lt}} \mathcal{F}_{\text{OT}}$ for the cases outlined in Theorem 11. Using the composition theorem (Theorem 6), we can replace \mathcal{F}_{CP} by an appropriate protocol in the \mathcal{F}_{CRS} hybrid model (e.g. Theorem 10) and the corollaries follow.

Secure Function Evaluation We can use a similar approach to construct (reactive) secure two-party function evaluation with long-term Rewinding UC security for one party. In the following we describe shortly the conceptual idea to highlight the adaptations.

In order to import a k -round UC-secure protocol into our framework, we need the committed-value oracle \mathcal{O}_{CCA} in the helper \mathcal{H} to be k -robust. For efficiency reasons, it is thus highly desirable to use building blocks with a constant round complexity. We propose a construction based on garbled circuits, which allows constant-round general two-party computation.

Most garbled circuit schemes only provide computational security. While constructions with information-theoretic security exist, they suffer from efficiency problems, especially as the circuit depth grows. Given the fact that we only can construct OT with long-term security for one party, the garbled circuit scheme, interestingly, does not need to provide information-theoretic security. Indeed, if the party P_i with information-theoretic security is known in advance (as we assume for the OT protocol), P_i performs the garbling and the OT protocol provides long-term security for P_{1-i} , then there is no information flow from P_{i-1} to P_i .

This shows that we can get a secure two-party function evaluation long-term-Rewinding UC security for one party based on garbled circuits. We leave further details for future work.