Karlsruhe Institute of Technology

Frameworks and Protocols for Composable Multi-Party Computation

Zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Jeremias Mechler

Tag der mündlichen Prüfung:	17. April 2023
Erster Referent:	Prof. Dr. Jörn Müller-Quade
Zweiter Referent:	Prof. Jesper Buus Nielsen, PhD

KIT – The Research University in the Helmholtz Association

www.kit.edu

I dedicate this thesis to my father, Wolfgang Mechler.

Acknowledgements

During my time at the Chair of Cryptography and Security, I was lucky enough to meet many great people who made the time leading up to my doctorate very enjoyable.

My greatest gratitude goes to Jörn Müller-Quade, who supervised this work and shared not only his knowledge, but also his joy in research over many years. At his chair, I have not only enjoyed the greatest academic and personal freedom, but have also been given the space for development in every respect. Jörn is my great role model when it comes to what a scientist should be like and what respectful interaction in professional life looks like.

It all started with Dirk Achenbach, who supervised my Bachelor's thesis and hired me as a student assistant. I have remained friends with him all these years, as well as with Bernhard Löwe, with whom I was able to work a short time later. Matthias Nagel, who co-supervised my Master's thesis and for whom no problem of any kind is too difficult, was also part of the team later on. I would like to thank them for the enjoyable time we spent together, especially when preparing the exhibition "Global Control and Censorship" at the ZKM!

A big thank you goes to Brandon Broadnax, who took me under his wings and taught me Universal Composability research. Without him, this thesis would not be what it is now. I also particularly enjoyed our chats about US politics.

Alexander Koch not only co-supervised my Master's thesis, but also co-authored many papers. No draft was good enough for Alex to not find room for improvement. I thank him very much for his time and knowledge shared over the years.

Without Michael Klooß and his knowledge of rewinding, which he generously shared with me, as well as his time, the second part of this work would not have been possible. Many thanks to Michael for bravely plunging into the depths of Universal Composability (and coming out unscathed)!

I would like to thank Rebecca Schwerdt for the lunches we shared, the chocolate we ate together, the tea we drank and her friendship in general.

Carmen Manietta, Holger Hellmuth and Willi Geiselmann have always been around since I was a student. Without you, nothing would have worked at the Chair of Cryptography and Security—so many, many thanks for your support.

I would also like to thank my other colleagues as well as my co-authors Lukas Beeck, Robin Berger, Dominik Doerner, Felix Dörre, Nico Döttling, Matthias Gabel, Roland Gröll, Timon Hackenjos, Sven Maier, Anne Müller, Tobias Nilges, Markus Raiber, Gunter Schiefer, Astrid Ottenhues, Jochen Rill and Marcel Tiepelt.

Another thank you goes to everyone involved in the doctoral process for their participation, especially to Jesper Buus Nielsen, who kindly served as second reviewer.

Finally, I would like to thank my wonderful parents.

Abstract

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, even if a subset of the parties is corrupted. Due to strong feasibility results, it is known that MPC protocols exist for almost every efficiently computable task, requiring only authenticated point-to-point communication, but no centrally trusted entity.

In today's highly interconnected world, one is often faced with a situation where not only one instance of a single protocol is executed, but several different protocols are executed at the same time. In such a setting, it is necessary that protocols are secure under *composition*, *i.e.* even when many instances of possibly different protocols are executed at the same time.

Building upon the established notion of Universal Composability (UC) due to Canetti (FOCS 2001), in this thesis we present frameworks and protocols for composable multi-party computation with novel properties and security guarantees.

Environmentally Friendly Composable Multi-Party Computation in the Plain Model from Standard (Timed) Assumptions

If a protocol fulfills the notion of UC security, its security holds under *universal composition*, *i.e.* even when it is executed together with arbitrary other protocols in a concurrent setting. While this guarantee is very strong, UC security faces a major shortcoming: In order for a protocol to be UC-secure, its parties need access to a so-called *trusted setup*, *i.e.* an entity that provides a certain functionality like a public key infrastructure or a common reference string (CRS). In a setting where parties distrust each other, *i.e.* the very setting of MPC, it is often unclear who could be trusted to provide such a setup. This practical problem is exacerbated by a theoretical one, namely the fact that setups substantially and artificially weaken a protocol's security: If the setup is corrupted, all security may be lost. This introduction of a single point of failure by UC security goes against the very promise of MPC, which is for parties to need very little or even no trust in others.

In order to reconcile the conflict between achieving composability on the one hand and not needing strong trust assumptions on the other hand, it is highly desirable to investigate alternative security notions for composable MPC that can be realized in the *plain model*, *i.e.* assuming only authenticated communication and possibly cryptographic hardness assumptions, but without setups.

While many approaches for composable MPC in the plain model exist, they provably cannot achieve the same properties as UC security. One major drawback of previous solutions is that they may negatively affect the security of protocols executed alongside, *i.e.* in the "environment", which is not the case with UC security. Canetti *et al.* (FOCS 2010, FOCS 2013) proposed the notion of *environmental friendliness*, which allows to judge this negative impact. Informally, limited environmental friendliness comes from the way composability is achieved without a setup, namely by performing superpolynomial-time computations.

The first contribution of this thesis addresses this problem of limited environmental friendliness by providing a new notion for composable MPC in the plain model along with protocols fulfilling this notion.

The new notion is the first one for composable MPC in the plain model that does not negatively affect *any* polynomial-time security property of protocols executed concurrently or previously, *i.e.* the first to achieve *full environmental friendliness*. Also, it is the first to use timed assumptions, *i.e.* assumptions that only hold against adversaries with a certain runtime bound, to achieve composability. In particular, we use *timed commitment schemes* whose hiding property is only guaranteed for a certain time. The key idea is to use efficient, *i.e.* polynomial-time, simulation techniques, which do not affect the security of other protocols. This is in contrast to previous notions for composable multi-party computation in the plain model adhering to the *simulation paradigm*, which require inefficient computations.

As UC security has no notion of time, it is unsuitable for protocols using timed assumptions. We suitably adapt it such that parties can set up timers counting the computational steps performed in the current execution and check their expiry, thus allowing to leverage the security provided by timed cryptographic assumptions. The resulting notion is called *Time-Lock UC (TLUC)* security and features constant-round general composable multi-party computation from well-studied and relatively weak (*i.e.* standard) polynomial-time and timed assumptions in the plain model.

Moreover, our constructions do not require inefficient *non-black-box techniques*, but use their building blocks via their input-output interface only. Also, the simulation does not rely on inefficiently obtained *advice* that can be used to solve certain (uniformly) hard problems, but may negatively affect the security of previously started protocols. These important properties are not achieved together by previous notions for composable MPC in the plain model.

We are also the first to enable the reuse of arbitrary UC-secure protocols. Thus, one can often take the best UC-secure protocol for a given task and realize the setup in TLUC, yielding a TLUC-secure protocol in the plain model with a very small overhead for a large class of setups.

Updatable Composable Security

Orthogonal to the necessity of trusted setups for UC security is the fact that (general) secure multi-party computation can only be performed under certain *assumptions*.

Examples include the existence of communication facilities, *e.g.* point-to-point authenticated communication, the existence of an honest majority, or, more importantly, the existence of hard problems.

Of course, an assumption made now may turn out to be wrong in the future—either, for example, because it was wrong in the first place, or because the circumstances justifying the assumption have changed. While this problem is well-understood in the case of corruptions, where the concept of *adaptive corruptions* deals with a setting where initially honest protocol parties are becoming dishonest during the execution of a protocol, there is no general analogue concept for cryptographic hardness assumptions.

Given the insecurity of widely-used cryptographic building blocks such as DES, MD5 or SHA-1, the lack of a framework for analyzing the security of protocols in such a setting poses a major problem, as it is currently impossible to adequately model and quantify the consequences of a hardness assumption becoming invalid. With the future availability of universal quantum computers, these problems will become exacerbated as many commonly used hardness assumptions such as RSA will provably become insecure, along with protocols using them.

Long-term Security, introduced by Müller-Quade and Unruh (TCC 2007, JoC 2010), extends UC security to a setting where polynomial-time hardness assumptions hold during the execution of a protocol, but *all* hardness assumptions become invalid after the execution has finished. While this notion already provides much stronger guarantees than UC security in a setting where hardness assumptions may become invalid, it is inadequate if this loss may already occur during protocol execution.

Given that there are protocols that can provide meaningful security in such a setting because they can be *updated* to new hardness assumptions, this shortcoming is not only of theoretical nature. A very important and natural example is the computational binding property of a statistically hiding commitment scheme that can be preserved by (consistently) re-committing to the same value. Currently, there is no framework that allows to analyze the composable security of such updatable commitment schemes and protocols that can be built from them.

Unfortunately, the setting of updatable composable security is subject to the same strong impossibility results as long-term security: Commitment schemes that are, at the same time, composable, statistically hiding and computationally binding, cannot be constructed solely from *e.g.* a CRS.

Our contribution towards solving this problem is two-fold: First, we circumvent the impossibility results due to Müller-Quade and Unruh by using new techniques that allow rewinding-based simulation in a way that universal composability is possible. As a result, we are the first to construct a commitment scheme in the CRS-hybrid model that is simultaneously statistically hiding, computationally binding and composable. For this, we again only require well-understood and relatively weak hardness assumptions. Moreover, the round efficiency is asymptotically optimal. Using this commitment scheme, we can construct composable zero-knowledge as well as composable commit-and-proof with long-term security.

Second, we extend this statistically hiding commitment scheme such that its computational binding property can be repeatedly updated by re-committing in a consistent way. We propose an ideal functionality for updatable commitment schemes and prove that our protocol realizes this functionality under *Updatable UC security*, our new security notion that extends UC security and long-term security such that cryptographic hardness assumptions can be invalidated before, during and after protocol execution. Again, we are able to re-use large classes of UC-secure protocols.

We also prove an impossibility result, namely that oblivious transfer with long-term security for *both* parties is impossible in the setting we consider. On the positive side, we give a construction for composable oblivious transfer with long-term security for *one* party, which is provably the best security guarantee. We also sketch how this protocol can be used to achieve composable general two-party computation with long-term security for *one* party. These positive and negative results thus give a complete characterization of composable two-party computation with long-term security for the setting we consider.

Zusammenfassung

Eine sichere Mehrparteienberechnung (kurz MPC für *multi-party computation*) erlaubt es Parteien, die sich gegenseitig misstrauen, Berechnungen auf ihren geheimen Eingaben durchzuführen. Dabei werden Eigenschaften wie Korrektheit, Vertraulichkeit oder die Unabhängigkeit der Eingaben garantiert – selbst dann, wenn eine Teilmenge der Parteien korrumpiert ist. Aufgrund starker Möglichkeitsergebnisse ist bekannt, dass es MPC-Protokolle für fast jede effizient berechenbare Aufgabe gibt. Dafür ist lediglich authentifizierte Punkt-zu-Punkt-Kommunikation erforderlich, aber insbesondere keine Entität, der alle Parteien vertrauen.

In der heutigen stark vernetzten Welt wird oftmals nicht nur eine Instanz desselben Protokolls ausgeführt, sondern vielmehr verschiedene Protokolle gleichzeitig. Deshalb ist es notwendig, dass Protokolle auch unter *Komposition* sicher sind, also selbst dann, wenn viele Instanzen von möglicherweise verschiedenen Protokollen gleichzeitig ausgeführt werden.

Aufbauend auf dem etablierten Begriff der *Universal Composability* (UC) von Canetti (FOCS 2001) präsentieren wir in dieser Dissertation Frameworks und Protokolle für komponierende sichere Mehrparteienberechnungen mit neuartigen Eigenschaften und Sicherheitsgarantien.

Environmentally Friendly Composable Multi-Party Computation in the Plain Model from Standard (Timed) Assumptions

Wenn ein Protokoll den Begriff der UC-Sicherheit erfüllt, gilt seine Sicherheit unter *universeller Komposition*, also selbst dann, wenn es zusammen mit beliebigen anderen Protokollen gleichzeitig (englisch *concurrently*) ausgeführt wird. Auch wenn diese Garantie sehr stark ist, hat UC-Sicherheit doch ein erhebliches Manko: Damit ein Protokoll UC-sicher sein kann, müssen die Protokollparteien auf ein so genanntes *vertrauenswürdiges Setup* zugreifen können, also eine Entität, die eine gewisse Funktionalität wie eine Public-Key-Infrastruktur oder einen *common reference string* bereit stellt. Wenn sich die Parteien gegenseitig misstrauen, also genau der bei MPC betrachteten Situation, ist es oftmals unklar, wer vertrauenswürdig genug sein könnte, um das Setup bereitzustellen. Dieses praktische Problem wird durch ein theoretisches verschärft, denn Setups schwächen die Sicherheit eines Protokolls substanziell und auf unnatürliche Weise: Ist das Setup korrumpiert, so können alle Sicherheitsgarantien eines Protokolls verloren gehen. Diese Einführung eines *single point of failure* steht dem Versprechen von MPC, anderen wenig oder auch gar nicht vertrauen zu müssen, diametral entgegen.

Um den Konflikt zwischen dem Erreichen von Komponierbarkeit einerseits und dem Verzicht auf starke Vertrauensannahmen andererseits aufzulösen, ist es höchst wünschenswert, alternative Sicherheitsbegriffe für komponierende Mehrparteienberechnungen zu untersuchen. Interessant sind insbesondere solche, die im *plain model*, also nur unter der Annahme von authentifizierter Kommunikation und möglicherweise kryptographischer Komplexitätsannahmen erreicht werden können, aber keine Setups benötigen.

Auch wenn es viele Ansätze für komponierende Mehrparteienberechnungen im *plain model* gibt, so können diese beweisbar nicht dieselben Eigenschaften wie UC-Sicherheit erreichen. Ein großer Nachteil bisheriger Lösungen ist, dass diese die Sicherheit von Protokollen, die nebenläufig, also in der „Umgebung", ausgeführt werden, negativ beeinflussen können, was bei UC-Sicherheit nicht der Fall ist. Canetti *et al.* (FOCS 2010, FOCS 2013) haben den Begriff der *environmental friendliness* vorgeschlagen, der es erlaubt, diesen negativen Einfluss zu beurteilen. Die eingeschränkte *environmental friendliness* kommt etwa durch die Art und Weise, wie Komponierbarkeit ohne Setup durch eine Superpolynomialzeit-Simulation erreicht wird, bedingt.

Der erste Beitrag dieser Dissertation befasst sich mit diesem Problem der eingeschränkten *environmental friendliness*. Dazu wird ein neuer Begriff für komponierende Mehrparteienberechnungen im *plain model* vorgestellt, zusammen mit Protokollen, die den Begriff erfüllen.

Der neue Begriff ist der erste, der Sicherheitseigenschaften (gegenüber Polynomialzeitangreifern) von anderen Protokollen, die nebenläufig ausgeführt werden oder deren Ausführung geendet hat, nicht negativ beeinflusst, also der erste, der *volle environmental friendliness* erreicht. Er ist ebenso der erste, der *timed* kryptographische Annahmen einsetzt, also Annahmen, die nur gegenüber Angreifern mit einer bestimmten Laufzeitschranke gelten, um Komponierbarkeit zu erreichen. Wir verwenden insbesondere *timed* Commitment-Verfahren, deren Hiding-Eigenschaft nur für eine gewisse Zeit garantiert ist.

Die wesentliche Idee ist die Verwendung von effizienten, also in Polynomialzeit ausführbaren, Simulationtechniken, die die Sicherheit anderer Protokolle nicht beeinflussen. Dies steht im Gegensatz zu vorherigen Sicherheitsbegriffen für komponierende Mehrparteienberechnungen im *plain model*, die dem Simulationsparadigma folgen und dabei ineffiziente Berechnungen benötigen.

Da UC-Sicherheit keinen Zeitbegriff umfasst, ist sie für Protokolle, die *timed* Annahmen verwenden, ungeeignet. Wir passen den UC-Sicherheitsbegriff so an, dass Parteien Timer aufsetzen können, die die Berechnungsschritte in der aktuellen Ausführung zählen. Anschließend können die Parteien prüfen, ob diese Timer abgelaufen sind. So können die Sicherheitsgarantien von *timed* Annahmen genutzt werden. Der resultierende Begriff heißt *Time-Lock-UC-Sicherheit* (kurz *TLUC-Sicherheit*) und ermöglicht allgemeine Mehrparteienberechnungen mit konstanter Rundenanzahl basierend auf sehr gut verstandenen und vergleichsweise schwachen kryptographischen Annahmen (so genannten Standardannahmen) zusammen mit *timed* kryptographischen Annahmen im *plain model*.

Darüber hinaus benötigen unsere Konstruktionen keine ineffizienten so genannten Non-Black-Box-Techniken, sondern nutzen Bausteine nur über ihre Ein- und Ausgabe-Schnittstellen. Auch benötigt unsere Simulationstechnik im Gegensatz zu anderen

Ansätzen keinen ineffizienten *advice*, der helfen kann, bestimmte schwierige Probleme zu lösen, dabei aber möglicherweise die Sicherheit von bereits gestarteten Protokollinstanzen negativ beeinflusst. Diese wichtigen Eigenschaften werden von bisherigen Sicherheitsbegriffen für komponierende Mehrparteienberechnungen im *plain model* nicht gemeinsam erreicht.

Weiterhin ist unser Ansatz der erste, der die Wiederverwertbarkeit von beliebigen UC-sicheren Protokollen erlaubt. So ist es oftmals möglich, das Setup des besten UC-sicheren Protokolls für eine bestimmte Aufgabe in TLUC zu realisieren. Das resultierende Protokoll ist TLUC-sicher im *plain model*, wobei der zusätzliche Aufwand für eine große Klasse von Setups sehr gering ist.

Updatable Composable Security

Orthogonal zur Notwendigkeit von Setups zum Erreichen von UC-Sicherheit ist die Tatsache, dass sichere Mehrparteienberechnungen nur unter bestimmten *Annahmen* möglich sind. Beispiele hierfür sind das Vorhandensein einer Kommunikationsinfrastruktur wie authentifizierter Punkt-zu-Punkt-Kommunikation, die Existenz einer ehrlichen Mehrheit oder, noch wichtiger, der Existenz von schwierigen Problemen.

Selbstverständlich kann sich eine heute getätigte Annahme morgen als falsch erweisen – entweder, weil sie beispielsweise von vornherein falsch war, oder weil sich die Umstände, die die Annahme gerechtfertigt haben, geändert haben. Dieses Problem ist im Falle von Parteienkorruption mit dem Konzept der *adaptiven Korruption* sehr gut verstanden. Dabei wird abgebildet, dass Protokollparteien zunächst ehrlich sind, aber im Laufe der Protokollausführung korrumpiert werden können. Ein analoges Konzept für kryptographische Komplexitätsannahmen gibt es jedoch nicht.

In Anbetracht der Unsicherheit von breit eingesetzten kryptographischen Bausteinen wie DES, MD5 oder SHA-1 ist das Fehlen eines Frameworks, mit dem die Sicherheit von Protokollen in einer solchen Situation analysiert werden kann, ein großes Problem – im Moment ist es nämlich nicht möglich, die Folgen des Ungültigwerdens einer Komplexitätsannahme zu modellieren oder zu quantifizieren. Mit der zukünftigen Verfügbarkeit von universellen Quantencomputern verschärft sich das Problem noch, da viele weithin eingesetzte kryptographische Komplexitätsannahmen wie RSA beweisbar unsicher werden, zusammen mit den Protokollen, in denen sie verwendet werden.

Langzeitsicherheit (*long-term security*), eingeführt von Müller-Quade und Unruh (TCC 2007, JoC 2010) erweitert UC-Sicherheit dahingehend, dass während einer Protokollausführung Polynomialzeitannahmen gelten können, nach dem Ende der Ausführung aber *alle* Komplexitätsannahmen ihre Gültigkeit verlieren. Auch wenn dieser Sicherheitsbegriff viel stärkere Garantien als UC-Sicherheit bietet, wenn Komplexitätsannahmen ihre Gültigkeit verlieren können, ist er dennoch inadäquat, wenn der Gültigkeitsverlust schon während der Protokollausführung auftreten kann.

Bedenkt man, dass es Protokolle gibt, die in solch einem Fall sinnvolle Sicherheitsgarantien geben können, weil sie auf neue Komplexitätsannahmen „geupdated“ werden können, so ist dieses Manko nicht nur theoretischer Natur. Ein wichtiges und gleichzeitig natürliches Beispiel ist die Binding-Eigenschaft von Commitment-Verfahren, deren Hiding-Eigenschaft statistisch gilt: Die Binding-Eigenschaft kann durch (konsistentes) wiederholtes Committen auf denselben Wert erhalten werden. Im Moment gibt es kein Framework, mit dem die komponierende Sicherheit von solchen updatebaren Commitment-Verfahren sowie Protokollen, die man aus ihnen bauen kann, analysiert werden kann.

Leider unterliegt updatebare komponierende Sicherheit denselben starken Unmöglichkeitsergebnissen wie Langzeitsicherheit: Commitment-Verfahren, die gleichzeitig komponierend sind, deren Hiding-Eigenschaft statistisch und deren Binding-Eigenschaft komplexitätstheoretisch gilt, können nicht ausschließlich beispielsweise mithilfe eines *common reference string* konstruiert werden.

Unser Beitrag zur Lösung dieses Problems besteht aus den folgenden zwei Teilen: Erstens umgehen wir mithilfe von neuen Techniken, die eine Rewinding-basierte Simulation so ermöglichen, dass Universal Composability möglich ist, die Unmöglichkeitsergebnisse von Müller-Quade und Unruh. Damit sind wir die ersten, die ein Commitment-Verfahren im CRS-Hybridmodell konstruieren, für das die Hiding-Eigenschaft statistisch und die Binding-Eigenschaft komplexitätstheoretisch gilt und das gleichzeitig komponierend ist. Auch dafür benötigen wir nur sehr gut verstandene und vergleichsweise schwache kryptographische Komplexitätsannahmen. Darüber hinaus ist die Rundeneffizienz asymptotisch optimal. Unter Verwendung dieses Commitment-Verfahrens konstruieren wir komponierende Protokolle für Zero-Knowledge sowie Commit-and-Proof mit Langzeitsicherheit.

Zweitens erweitern wir das Commitment-Verfahren, dessen Hiding-Eigenschaft statistisch gilt, so, dass die komplexitätstheoretische Binding-Eigenschaft durch konsistentes wiederholtes Committen mehrmals geupdated werden kann. Wir stellen eine ideale Funktionalität für updatebare Commitments vor und beweisen, dass unser Protokoll diese Funktionalität relativ zu unserem neuen Sicherheitsbegriff der *Updatable-UC-Sicherheit* realisiert. Dabei erweitert Updatable-UC-Sicherheit die UC-Sicherheit und die Langzeitsicherheit so, dass kryptographische Komplexitätsannahmen vor, während und nach der Protokollausführung ungültig werden können.

Wir beweisen auch ein Unmöglichkeitsergebnis, nämlich die Unmöglichkeit von *oblivious transfer* mit Langzeitsicherheit für *beide* Parteien unter den von uns betrachteten Rahmenbedingungen. Positiv zu vermerken ist, dass wir eine Konstruktion für komponierenden *oblivious transfer* mit Langzeitsicherheit für *eine* Partei angeben können, was beweisbar die bestmögliche Sicherheitsgarantie ist. Ebenso skizzieren wir, wie dieses Protokoll dazu verwendet werden kann, um komponierende allgemeine Zwei-Parteien-Berechnungen mit Langzeitsicherheit für *eine* Partei durchzuführen. Diese Positiv- und Negativergebnisse stellen also eine vollständige Charakterisierung für komponierende Zwei-Parteienberechnungen mit Langzeitsicherheit in dem von uns betrachteten Umfeld dar.

Contents

1. Introduction

Whether it is our medical history, information about social interaction or financial transactions—almost every aspect of our lives leaves behind a digital footprint. If this data were available for analysis, society could gain many valuable insights towards solving its most important problems. At the same time, simply pooling this data for analysis poses a great threat due to its highly private nature, and better methods are needed.

A cryptographic tool enabling the private computation on sensitive data is a so-called *secure multi-party computation (MPC)*, which allows mutually distrusting parties to jointly perform distributed and decentralized computations on their secret inputs. Even if a subset of the parties is corrupted, *i.e.* jointly controlled by an adversary, protocols for MPC provide very strong security guarantees. In particular, participating in an MPC protocol traditionally does not allow a malicious party to learn anything about an honest party's secrets that it cannot compute from the computation's result and its own input. Moreover, additional guarantees such as *correctness* and *independence of inputs* are provided, all without needing to trust a central entity.

Due to strong feasibility results, (*e.g.* [Y86; GMW87; CLOS02]), general MPC constructions for (almost) every efficiently computable task are known [BKM$^+$21; DMM22; DKM$^+$22; MMN18]. Additionally, many protocols for special tasks exist (*e.g.* [DMM23; BMM21; BBK$^+$23; DKM$^+$22; AGH$^+$19]), providing for example better efficiency than a generic solution.

Examples for real-world problems that may be solved using MPC today include

- the intersection of private sets [DMM23], *e.g.* consisting of phone numbers for the purpose of contact discovery in messenger applications,
- contact tracing [BDH$^+$21] to warn if there have been close encounters with infected persons or
- a "similar patient query" [AHLR18], for example with the purpose of finding an appropriate donor of biological material, or
- computing the market clearing price [BCD$^+$09] for sugar beets.

These examples justify not only the theoretical interest in MPC, but also illustrate the growing practical relevance of cryptographic tools with strong privacy guarantees.

Defining Security. In order to capture (and prove) the security guarantees provided by a cryptographic protocol, a precise and mathematical definition of security is needed.

A popular and natural approach is the so-called *real-ideal paradigm* or *(stand-alone) real-ideal security* [G04]. With real-ideal security, the execution of a (real) protocol π in the presence of an adversary $\mathcal{A}$ (the "real execution") is compared to an execution of an ideal functionality $\mathcal{F}$ and an ideal-world adversary $\mathcal{S}$ that models attacks that can, in principle, not be ruled out (the "ideal execution").

In this ideal execution, all (honest) protocol parties directly interact with the ideal functionality $\mathcal{F}$, which is incorruptible and carries out the desired task by definition. First, all parties send their input to $\mathcal{F}$. The ideal-world adversary, also called the *simulator*, is responsible for providing the inputs of the corrupted parties. After all parties have provided their input, $\mathcal{F}$ performs the specified computation. The simulator receives the outputs for the corrupted parties, while the honest parties directly receive their outputs from $\mathcal{F}$. Finally, the simulator outputs an arbitrary string of its choice.

The only possibility for the ideal-world adversary to influence the computation is by providing an input of its choice for the corrupted parties and (usually) by suppressing the output of a subset of the honest parties, possibly depending on the output of the corrupted parties. Clearly, in the ideal execution, properties such as *correctness*, *privacy* or *independence of inputs*, *i.e.* the guarantee that the input of a corrupted party does not depend (possibly only through a relation) on the input of an honest party, are satisfied.

In the real execution, the protocol π is executed. The adversary $\mathcal{A}$ is responsible for the delivery of messages, usually being allowed to suppress, but not to change or inject messages from honest parties. Also, $\mathcal{A}$ controls the corrupted parties. At the end of the execution, the honest parties output the results of the computation.

In order to prove the security of a protocol π, one shows that it realizes an appropriate ideal functionality $\mathcal{F}$ that captures the desired security guarantees. To this end, it is necessary to prove the existence of a simulator such that the probability ensembles consisting of

- the outputs of the honest parties in the real execution as well as the view of the adversary in the real execution resp.

- the outputs of the honest parties in the ideal execution as well as the output of the simulator

are indistinguishable.

If this holds, it is easy to see that all guarantees that hold in the ideal execution carry over to the real execution—otherwise, they would not be indistinguishable.

In order to perform the simulation, the simulator typically executes π and $\mathcal{A}$ "in its head" in an imaginary execution. (Because of this "simulation in the head" of the real execution, the ideal-world adversary is called the simulator.) At the same time, it interacts with the ideal functionality $\mathcal{F}$. In the execution "in its head", the simulator must faithfully simulate the real execution for the adversary $\mathcal{A}$ by simulating the honest parties, albeit without knowing their inputs and outputs[1]. Also, the simulator must provide the ideal functionality with the inputs of the corrupted parties. These secret inputs must be *extracted* by $\mathcal{S}$ through interaction with the internally emulated adversary $\mathcal{A}$. Perhaps surprisingly, all this is possible without knowing the honest parties' inputs and outputs if the protocol is appropriately designed—even if the adversary behaves like an honest party.

At first glance, this requirement (and ability of the simulator) seems to contradict the privacy requirement, as merely interacting with a party (corrupted or not) should not enable one to learn its secrets. Indeed, if the real-world adversary could, like the simulator, learn a (honest) party's input by interacting with it, there would be no security.

However, the simulator has an *advantage* over the real-world adversary and over honest parties: It may *rewind* the execution it simulates "in its head", *i.e.* reset this simulated execution to an earlier state, send different messages and possibly observe different answers. Clearly, this capability makes the simulator very powerful. At the same time, it is not available to the real-world adversary, leading to a meaningful security guarantee.

Using real-ideal security, security of important tasks such as *secure function evaluation*, *oblivious transfer* or *zero-knowledge proof systems* can be captured.

Composable Security. Unfortunately, stand-alone real-ideal security is not closed under composition, *i.e.* when multiple instances of a protocol are executed at the same time. As a consequence, *all security* of a protocol π previously proven secure may be lost even when only two instances of *the same protocol* π are executed in parallel [GK90]. Clearly, in a highly connected world such as the Internet, where *arbitrary* protocols are executed *concurrently*, a security notion that is not closed under protocol composition is insufficient.

Thus, it is important to consider security notions that guarantee security under *universal composition*, *i.e.* in the presence of concurrent executions of the same or even other protocols that may be adversarially chosen with the intent to hurt the security of the protocol under analysis.

The established security notion for this setting is called Universal Composability (UC) [C01]. Similar to real-ideal security, the security of a protocol π is defined by the ideal functionality $\mathcal{F}$ it realizes. However, the experiment is extended to include a so-called *environment*, usually denoted by $\mathcal{Z}$, which serves as an interactive distinguisher, adaptively provides inputs to the parties and communicates with the adversary. By allowing arbitrary communication between the adversary and the environment, UC security naturally captures a setting where arbitrary other protocols are executed with an adversarial schedule alongside the protocol under analysis. As these other protocols are implicitly executed inside the environment, they do not need to be explicitly considered

[1] Depending on the function to be computed, some information about an honest party's secrets may be learned. For example, computing the bit-wise exclusive-or y of two values x_1 and x_2 *always* allows to reconstruct x_2 from y and x_1 and vice versa.

during the security proof. As a consequence, protocols that satisfy UC security remain secure under *general concurrent* or *universal* composition.

Despite these strong and desirable security guarantees, UC security is lacking in several aspects. In the following, we discuss two major shortcomings.

Shortcoming: Necessity of Setups. The strong guarantees of UC security come with a price to pay. While real-ideal security can be achieved in the *plain model*, *i.e.* assuming authenticated communication only, there are well-known and strong impossibility results with respect to UC security [CF01; CKL03; PR08; KL11]: Unless an honest majority exists, protocols with natural properties and UC security (against malicious adversaries) necessitate the use of a so-called *trusted setup* like a common reference string (CRS), a public key infrastructure (PKI), a random oracle (RO) or tamper-proof hardware tokens [MMN18].

With (stand-alone) real-ideal security, the extraction of inputs of corrupted parties was possible by rewinding the adversary "in the head" of the simulator. In particular, the adversary only interacted with the "outside world" twice when receiving its input and giving its output. With UC security, rewinding the adversary "in the head" of the simulator is not possible anymore: In contrast to the adversary in the real-ideal paradigm, the UC adversary may communicate with the "outside world", *i.e.* the environment, at *any time*. Apart from technicalities that prevent the simulator from also simulating the environment "in its head" in order to rewind it together with the adversary, such a notion might not provide meaningful guarantees whatsoever, as *all* protocols executed concurrently would be rewound too. Instead, UC security requires so-called *straight-line simulation*, *i.e.* simulation without rewinding other machines.

To this end, UC-secure protocols need to be designed in a way that enables extraction when the setup is, for example, modified to contain a *hidden backdoor*. In the ideal execution, this can be done by the simulator, as it provides the setup. As each setup is used for a single protocol instance only, the use of setups ensures independence between different protocol sessions, enabling universal composability in the first place.

However, having to use setups is undesirable for a number of reasons. From a theoretical perspective, introducing a setup that *e.g.* allows the extraction of secrets introduces not only an arguably artificial vulnerability, but also a single point of failure. If a setup is corrupted, *e.g.* because it is provided by a dishonest entity, all security may be lost. Given the fact that *non-composable* MPC can be achieved without trusted setups, this is highly unsatisfactory.

From a practical perspective, there is also the unsolved question of which entity can be trusted enough to provide a setup. If a multi-party computation is done because the participating parties do not trust each other, an entity trusted enough to provide the setup might just not exist.

As a consequence, numerous notions for composable MPC in the plain model have been investigated (*e.g.* [P03; BS05; LPV09; GGJS12; GKP18; DMRV13; PS04; CLP10; CLP13a; BDH$^+$17]). Since they do not rely on trusted setups, they are provably weaker than UC security. In particular, they suffer from the problem of negatively affecting the security of protocols that are executed alongside or have started before. This is due to the fact that the simulator needs an advantage over the real-world adversary: If the ability to perform rewinding is ruled out and there are no setups, one typically provides the simulator with a computational advantage, allowing it to (indirectly) perform inefficient, *e.g.* super-polynomial-time or non-uniform, computations. As the powerful simulator may now violate assumptions made to prove the security of other protocols executed alongside, they could become totally insecure.

Shortcoming: Indefinite Validity of Cryptographic Hardness Assumptions. Hard problems are provably necessary for certain tasks, *e.g.* general MPC without an honest majority or a very strong setup assumption. Often overlooked is the fact that problems that are (believed to be) hard today may not be hard tomorrow: Either, the problem may not have been hard in the first place and was only wrongfully assumed to be so, or algorithmic or computational advancements have made the problem tractable. Notable examples for the first case include *e.g.* cryptographic hash functions like MD5 [K06] or SHA-1 [SKP16], while the encryption scheme DES, which remains structurally unbroken but suffers from short keys, is a good example for the second case.

When hardness assumptions used in a protocol become invalid, the consequences may be manifold: If the protocol in question has finished, secret inputs and outputs of (honest) parties may leak. If the protocol is still in progress, additional security properties may be lost, *e.g.* the independence of inputs or the soundness of proof systems.

Unfortunately, the UC framework is unsuitable for analyzing the security of a protocol in such a setting as there is no mechanism that captures hardness assumptions becoming invalid at some point.

Long-term security introduced by Müller-Quade and Unruh [MU10] extends the notion of UC security to consider a setting where *all* hardness assumptions become invalid after a protocol execution has finished. As such, long-term security is suitable to analyze *e.g.* privacy guarantees of protocols that use RSA and terminate before universal quantum computers become available. In order to satisfy long-term security, the environment's view in the real execution must be statistically indistinguishable from its view in the ideal execution.

Surprisingly, protocols satisfying this strong and desirable security notion can be constructed, albeit using strong (and arguably impractical) setups like trusted signature cards. At the same time, Müller-Quade and Unruh [MU10] prove that the natural class of *long-term-revealing* setups, *e.g.* common reference strings or certain classes of public key infrastructures, cannot be used (on their own) to construct composable commitment schemes with long-term security, *i.e.* commitment schemes that statistically hide the value committed to after protocol has finished. Given that these setups are sufficient for standard UC security, this result may seem surprising.

Still, long-term security is inadequate to analyze the security of protocols in a setting where some hardness assumptions may already become invalid during protocol execution, while some remain valid at the same time, possibly beyond the end of the execution.

For example, a protocol for instant messaging used today may have a very long runtime, going beyond the point when universal quantum computers are available.

Not having an appropriate (*i.e.* general and composable) security notion for such a setting is very unfortunate, as some building blocks naturally admit the ability to *update* to a new hardness assumption. If this update is performed in time, security may be preserved. Examples include commitment schemes with stand-alone security, a statistical hiding property and a computational binding property that can be updated [DL15; CDG$^+$15; BGB17] in some way. The security of such an updatable commitment scheme can neither be analyzed with standard UC security nor with long-term security. The former fails to capture the possibility of hardness assumptions becoming invalid completely, the latter is both too optimistic and pessimistic in the sense that *all* hardness assumptions remain valid throughout the protocol execution, but become invalid immediately after.

Contribution of the Thesis

The contribution of this thesis consists of new frameworks and protocols for composable MPC that address the above shortcomings.

Environmentally Friendly Composable Multi-Party Computation in the Plain Model from Standard (Timed) Assumptions [BMM21]

In order to achieve composable security, the simulator needs an advantage over the environment. In the case of UC security, this advantage is the ability to simulate the setup. For example, the simulator may embed a trapdoor into a common reference string in a way that is undetectable for the environment.

In contrast, composable security in the plain model is usually achieved by using a super-polynomial-time simulation, *e.g.* by either allowing simulators having super-polynomial runtime complexity or by giving the polynomial-time simulator access to super-polynomial-time resources. The inherent drawback of such approaches is that super-polynomial-time simulation affects other polynomial-time protocols running concurrently. The consequences are two-fold: i) A protocol π realizing some functionality $\mathcal{F}$ under super-polynomial-time simulation cannot be plugged into arbitrary UC protocols ρ in the $\mathcal{F}$-hybrid model while retaining security of the composed protocol. ii) The simulation may not be *environmentally friendly*, *i.e.* it may negatively affect the security of polynomial-time protocols running alongside.

Our first contribution is a new approach for composable MPC in the plain model that fully addresses these problems. To this end, we propose a dual approach that is, to the best of our knowledge, completely novel: Instead of giving a super-polynomial advantage to the simulator, we *temporarily* restrict the adversary and the environment. These temporary restrictions allow the use of timed primitives, *i.e.* primitives that are not secure against probabilistic polynomial-time (PPT) adversaries, but only against adversaries adhering to a certain runtime bound. This is a first in the context of composable security in the plain model.

In more detail, we allow parties to set up *timers* parameterized by a number of steps t and require the environment to obey these timers, *i.e.* correctly signal if the timer has *expired* because the experiment has performed t or more steps. In our constructions, we use timers to "protect" the timed hiding property of timed commitment schemes.

Informally, a commitment scheme allows a *committer* to create a *commitment* to a secret. Before the commitment is *opened* by the committer, the committed secret remains *hiding* for every malicious (PPT) *receiver*. Conversely, commitment schemes also feature the *binding* property, guaranteeing that a malicious (PPT) committer is committed to its secret, *i.e.* cannot change it to a different value. Timed commitment schemes relax the hiding property, only guaranteeing it against adversaries that adhere to a certain runtime bound t. Additionally, it is usually guaranteed that a commitment can be *extracted* within $T > t$ steps, where T is polynomial. This property is not implied by the timed hiding property. Also, it clearly does not hold for (non-timed) commitment schemes, where the hiding property is guaranteed against *every* polynomial-time adversary.

By setting up appropriate timers, we ensure that timed commitments where the receiver is corrupted *initially* remain hiding for the environment. At some point, the environment will be able to learn the secret protected by the timed commitment. Conversely, the simulator, which is not required to obey timers, can extract timed commitments created by the environment. Surprisingly, this weak and only *temporary* advantage is sufficient to set up a long-lived trapdoor based on standard polynomial-time hardness assumptions, enabling composability.

These fine-grained asymmetries are not captured by previous notions for composable security. We thus introduce the notion of *Time-Lock UC (TLUC)* security, which is a variant of UC security that allows the use of building blocks with timed security properties. As TLUC simulators run in strict polynomial time, TLUC-secure protocols are *environmentally friendly* to all game-based properties against polynomial-time adversaries of protocols running alongside.

In previous approaches, the simulator kept its advantage throughout and beyond the protocol execution. In our setting, this is not the case anymore. Indeed, the environment will eventually be able to break *all* timed assumptions and thus *e.g.* learn the values of *all* timed commitments. We thus had to come up with a novel simulation technique that allows the simulator to set up a trapdoor while preventing the environment, which will eventually be just as powerful as the simulator, to do the same. At the same time, the environment that will eventually "catch up" on the runtime advantage of the simulator, must not be able to notice when the simulator has deviated from the protocol and set up a trapdoor.

Combining stand-alone timed commitment schemes with stand-alone non-malleable commitment schemes, we construct a commitment scheme that concurrently realizes many instances of $\mathcal{F}_{\mathrm{COM}}$, the ideal functionality for commitments, under TLUC security in the plain model.

Theorem 1.1 (Composable Commitments in the Plain Model, informal)**.** *If trapdoor PRGs with dense public description, perfectly binding homomorphic commitment schemes* *and timed commitment schemes exist, then there exists a black-box constant-round commitment scheme in the plain model that concurrently TLUC-realizes $\mathcal{F}_{\mathrm{COM}}$.*

As TLUC simulation is strictly polynomial-time, it does not "hurt" the security of other polynomial-time protocols running concurrently. This allows us to reuse arbitrary UC protocols without loss of security. In particular, we can take a UC-secure protocol ρ that uses one instance of an ideal functionality $\mathcal{F}$ to realize an ideal functionality $\mathcal{G}$ with UC security, realize $\mathcal{F}$ within TLUC in the plain model with a protocol π and plug π into ρ. Then, it follows that the composed protocol $\rho^{\mathcal{F}\to\pi}$, *i.e.* ρ where the single instance of $\mathcal{F}$ has been replaced with the protocol π, TLUC-realizes the ideal functionality $\mathcal{G}$.

In the notation of this thesis, which will be introduced later on, this reusability of UC-secure protocols is captured in the following proposition.

Proposition 1.1 (UC Reusability)**.** *Let ρ, σ and ϕ be PPT protocols such that ρ makes one subroutine call to ϕ and such that ρ UC-emulates σ. Let π be a PPT protocol such that π TLUC-emulates ϕ. Then, $\rho^{\phi\to\pi}$ TLUC-emulates σ.*

In particular, the UC reusability often allows to "bootstrap" the *best* UC-secure protocol ρ for a specific task by realizing the setup of ρ within TLUC in the plain model. In many cases, *e.g.* if the setup of ρ is a CRS, the resulting overhead is (asymptotically) very small and, in particular, independent of the inputs of the parties in ρ. We stress that this important property is *not* provided for arbitrary protocols ρ by previous notions for composable MPC in the plain model.

Using an appropriate general MPC protocol in the $\mathcal{F}_{\mathrm{COM}}$-hybrid model, we can achieve composable general TLUC-secure MPC in the plain model. To this end, we can state a constant-round protocol $\pi_{\mathcal{F}}^{BB}$ for (almost) every ideal functionality $\mathcal{F}$ such that $\pi_{\mathcal{F}}^{BB}$ concurrently TLUC-realizes $\mathcal{F}$.

Theorem 1.2 (Constant-Round MPC in the Plain Model, informal)**.** *If timed commitment schemes and perfectly binding homomorphic commitment schemes as well as enhanced trapdoor permutations with dense public descriptions exist, then for every well-formed functionality $\mathcal{F}$, there exists a constant-round black-box protocol $\pi_{\mathcal{F}}^{BB}$ in the plain model such that $\pi_{\mathcal{F}}^{BB}$ concurrently TLUC-realizes $\mathcal{F}$.*

The resulting protocol is environmentally friendly.

Updatable Composable Security

As our second contribution, we present a framework that allows to analyze the security of protocols in a setting where cryptographic hardness assumptions may become invalid.

In the UC framework, all entities run in probabilistic polynomial-time (PPT) and keep their runtime complexity throughout the execution. In order to allow environment and adversary to adaptively gain additional computational powers, we use the general mechanism proposed by [PS04; CLP10] and introduce a so-called *helper* that captures a number of stateless and deterministic *complexity oracles*. Initially disabled, a complexity oracle can be enabled by the environment at any time and then allows environment and adversary to perform certain (possibly inefficient) computations, *e.g.* to break an instance of the RSA problem. Ideal functionalities may query which complexity oracles are activated and change their behavior accordingly.

In such a setting where cryptographic hardness assumptions may become invalid, it is highly desirable to provide information-theoretic security guarantees to at least a subset of protocol parties. Unfortunately, our setting is subject to the impossibility results of Müller-Quade and Unruh [MU10], ruling out *e.g.* a universally composable commitment scheme in the CRS-hybrid model with long-term security. As this impossibility result can be easily extended to protocols in the plain model (as long as straight-line simulation is used), previously established techniques that allow composable MPC in the plain model also cannot be easily used to achieve long-term security.

With the requirement of straight-line simulation being the culprit, we take a different approach and carefully allow the simulator to rewind the execution. With this power at hand, it can, for example, extract statistically hiding commitments, which is not possible in a straight-line way without a setup. As discussed previously, allowing rewinding in an UC-like execution may not provide meaningful security of other protocols if they are affected by the rewinding.

To this end, we take several precautions. First, instead of directly providing rewinding capabilities to the simulator, we outsource this capability to the helper that also provides the complexity oracle. This way, we guarantee that only a very specific rewinding technique is performed. For the used rewinding technique, we can show several properties, in particular the *k-robust quasi-PPT property* and the *k-robust composition-order invariance.* Using these properties, we can show that the rewinding does *not* negatively affect the security of protocols executed concurrently, as long as their (joint) round complexity is bounded by $O(k)$ rounds. To this end, we also prove that we fulfill *environmental friendliness* for k-round protocols.

While the general idea of introducing rewinding to the UC execution seems straight-forward, it is riddled with technical difficulties stemming from the fact that we have to deal with commitment schemes that are statistically hiding. For such commitment schemes, it is neither straight-forward to define the *value committed to* nor to use established security notions such as CCA security. Informally, this is due to the fact that extracting such commitments requires *black-box rewinding access* to the malicious committer (and possibly other entities that (indirectly) communicate with the malicious committer), which is not possible with an ordinary oracle. To solve this problem, we introduce the concept of *pseudo-oracles*, which generalizes oracles in the sense that a pseudo-oracle may have black-box rewinding access to its *caller* as well as possibly other machines as necessary.

At the same time, the rewinding techniques have to account for the possibility that the cryptographic hardness assumptions that enable the extraction through rewinding in the first place may become invalid at any point. In particular, we have to deal with the possible case that an assumption becomes invalid in a side-thread when rewinding, but not in the main thread of the execution.

Having established the necessary techniques and game-based security notions for commitment schemes, we incorporate them into the UC framework through the helper, requiring subtle changes to the UC execution experiment. We call the resulting framework the *Updatable UC framework* and the resulting notion *(Long-Term) Updatable UC security.*

With this helper that provides extraction capabilities, we construct a composable commitment scheme with long-term security in the CRS-hybrid model, circumventing the impossibility results due to Müller-Quade and Unruh [MU10]. In particular, its hiding property is guaranteed statistically, while its binding property is computational.

Theorem 1.3 (Long-Term-Secure Composable Commitment Scheme, informal)**.** *Let $\mathcal{O}_{\mathsf{CCA}}$ be a black-box committed-value (pseudo-)oracle. If there exists a commitment scheme* COM *that is CCA-binding and trapdoor w.r.t. $\mathcal{O}_{\mathsf{CCA}}$ and has an appropriate message space, then there exists a protocol that long-term-Updatable-UC-realizes the ideal functionality for commitments in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model.*

The construction requires standard assumptions only and can be instantiated based on a number of problems, *e.g.* the RSA problem, the DLOG problem or a lattice-based post-quantum assumption. Moreover, its round complexity is asymptotically optimal.

In a second step, we extend the above commitment scheme to feature an *updatable* binding property. This is achieved by re-committing to the secret as well as previous decommitments using a *typed commitment scheme*, *i.e.* a commitment scheme that can be can be instantiated using different hardness assumptions, but otherwise follows a common template. To define the security of this updatable commitment scheme, we have devised an ideal functionality for updatable commitments, which is provably realized by our updatable commitment scheme. To the best of our knowledge, we are thus not only the first to construct a composable updatable commitment scheme, but also the first to realize such a scheme with a statistical hiding property.

Theorem 1.4 (Long-Term-Secure Composable Updatable Commitment Scheme, informal)**.** *Let $\mathcal{O}_{\mathsf{CCA}}$ be a black-box committed-value (pseudo-)oracle and let $\mathcal{O}_{\mathsf{comp}}$ be a complexity oracle. If there exists a typed commitment scheme* COM *that is enhanced CCA-binding and enhanced trapdoor w.r.t. $\mathcal{O}_{\mathsf{CCA}}$ and $\mathcal{O}_{\mathsf{comp}}$ and has an appropriate message space, then there exists a protocol that long-term-Updatable-UC-realizes the ideal functionality for updatable commitments in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model.*

Next to the above feasibility results, we prove a new impossibility result: Even if rewinding is allowed, oblivious transfer with long-term security for *both* parties is provably impossible, unless very strong setups are used. This implies that general two-party computation with long-term security for both parties is impossible, too. Nevertheless, we construct composable oblivious transfer with long-term security for *one* party in the CRS-hybrid model. Given the impossibility result, this is the best one can hope for. We also sketch how this protocol for oblivious transfer can be used for composable (reactive) two-party computation with long-term security for one party. Thus, we give a full characterization of general two-party computation for our setting.

On the positive side, we show that certain tasks can be achieved with long-term security for both parties in the CRS-hybrid model. Examples include composable zero-knowledge or composable commit-and-proof.

Outline of the Thesis

Starting with Chapter 2, we present preliminaries that are relevant to the following chapters. In Chapter 3, we present our first result, namely a new notion and protocols for composable MPC in the plain model from standard (timed) assumptions with properties that have not been achieved before. In Chapter 4, we propose another variant of UC security dealing with the problem that cryptographic hardness assumptions may become invalid during the execution of a protocol. We present various protocols with long-term security, including a composable commitment scheme whose binding property can be updated to new cryptographic hardness assumptions. We conclude the thesis in Chapter 5 and give an outlook to future work.

2. Preliminaries

Unless noted otherwise, this chapter is based mainly based on [BMM21] with some parts taken from [BBK+23].

2.1. Notation

Let $n \in \mathbb{N}$. Then, $[n]$ denotes the set $\{1, \ldots, n\}$. Let $I \subseteq \{0,1\}^*$. Then, $|I|$ denotes the length of I. For $I \subseteq \{0,1\}^*$ and $i \in [|I|]$, $I[i]$ denotes the i-th bit of I. Let H_i be some hybrid. Then, $\mathrm{out}_i(\kappa, z)$ denotes the output of H_i on input $(1^\kappa, z)$. For the sake of better readability, we will often omit κ and z. $\mathsf{negl}(\kappa)$ denotes an unspecified negligible function in the security parameter $\kappa \in \mathbb{N}$. Similarly, $\mathsf{poly}(\kappa)$ denotes an unspecified polynomial function in κ. $x \overset{\$}{\leftarrow} Y$ denotes that x is drawn uniformly at random from the set Y. $x \leftarrow Y$ denotes that x is either the output of the probabilistic algorithm Y or sampled according to the probability distribution Y. ε denotes either the empty string or a (not necessarily negligible) function for the adversary's advantage.

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 $\Im(\mathcal{A}(x))$ for the image of $\mathcal{A}$ on input x. 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 the security parameter κ and z is some (possibly non-uniform) input depending on κ. We use the usual notation for probability ensembles and write $\overset{c}{\approx}$ for computational and $\overset{s}{\approx}$ statistical indistinguishability.

Often, we consider cryptographic protocols that are parameterized by *e.g. families* of message spaces $\{M_\kappa\}_{\kappa \in \mathbb{N}}$. For the sake of an easier notation, we usually refer only to *e.g.* a single message space M instead of the associated family.

For two interactive Turing 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 $\mathrm{out}_\mathcal{B}\langle \mathcal{A}(x), \mathcal{B}(y) \rangle (1^\kappa, z)$ (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 $\mathit{view}_\mathcal{A}$ for the view of party $\mathcal{A}$, which consists of the party's random tape, all its inputs and all messages it received and $\tau \leftarrow \langle \mathcal{A}, \mathcal{B} \rangle$ for the transcript of the interaction, *i.e.* the sequence of sent messages. Often, we also write $(\mathit{view}_\mathcal{A}, \mathit{view}_\mathcal{B}, \tau) \leftarrow \mathrm{out}\langle \mathcal{A}, \mathcal{B} \rangle$ to obtain the view of $\mathcal{A}$, the view of $\mathcal{B}$ as well as the transcript τ. By abuse of notation, we sometimes write $\langle \mathcal{A}, \mathcal{B} \rangle$ for a protocol. For a treatment of interactive Turing machines, see [G01].

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 4.3.2), and we write $\mathcal{A}^{\mathcal{O}}$ for the composed machine; an oracle $\mathcal{O}$ is itself a (potentially unbounded) machine.

2.2. Basic Concepts

We now introduce basic concepts, starting with negligible functions. Informally, a function is called negligible if it asymptotically approaches zero faster than any polynomial.

Definition 2.1 (Negligible Function, adapted from [G01]). We call a function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ *negligible* if for every positive polynomial $p(\cdot)$ there exists an N such that for every $n > N$, $|\mathsf{negl}(n)| < \frac{1}{p(n)}$.

Conversely, a function f is called *overwhelming* if $1 - f$ is a negligible function.

Next, we define probability ensembles as families of random variables that are usually indexed by a security parameter.

Definition 2.2 (Probability Ensemble [G01]). Let I be a countable index set. An *ensemble indexed by* I is a sequence of random variables indexed by I. Namely, any $X = \{X_i\}_{i \in I}$, where X_i is a random variable, is an ensemble indexed by I.

Using the above definition, we can define computational indistinguishability. Informally, two ensembles X and Y are computationally indistinguishable if the advantage of a computationally bounded distinguisher attempting to distinguish between X and Y is negligible. As we consider security against non-uniform adversaries throughout this work, we parameterize the probability ensembles accordingly.

Definition 2.3 (Computational Indistinguishability, adapted from [G01]). Two ensembles, $X = \{X_{n,z}\}_{n \in \mathbb{N}, z \in \{0,1\}^*}$ and $Y = \{Y_{n,z}\}_{n \in \mathbb{N}, z \in \{0,1\}^*}$ are *computationally indistinguishable* if for every probabilistic polynomial-time algorithm D, every positive polynomial $p(\cdot)$, every sufficiently large n and every $z \in \{0,1\}^*$, it holds that

$$|\Pr[D(1^n, X_{n,z}, z) = 1] - \Pr[D(1^n, Y_{n,z}, z) = 1]| < \frac{1}{p(n)}$$

If the above inequality is satisfied for every *unbounded* algorithm D, we say that the ensembles are *statistically indistinguishable*. If, additionally, the distinguishing advantage is zero, then the indistinguishability is *perfect*.

2.2.1. Qualities of Security Properties

Security properties of cryptographic building blocks often come in different qualities, *e.g.* with *computational*, *long-term*, *statistical*, *perfect* or *unconditional* security. We will use the following *informal* conventions:

- Computational security considers security against (possibly non-uniform) probabilistic polynomial-time (PPT) adversaries, which are admitted a negligible advantage. Usually (but not necessarily), cryptographic hardness assumptions are used to achieve computational security, possibly together with trusted setups such as a common reference string (CRS).

- Statistical (resp. perfect) security considers security against unbounded adversaries, which are admitted a negligible (resp. zero) advantage. Consequently, security cannot rely on computational hardness. However, it may still rely on model assumptions, such as a correctly distributed CRS, and hence is not *unconditional.*
- Long-term [MU10] or everlasting security constitutes a relaxation of statistical or perfect security: The adversary is split into a computationally bounded, say non-uniform PPT, "online" attacker $\mathcal{A}$ during the execution of a protocol. Then, $\mathcal{A}$ outputs an arbitrary string to an unbounded distinguisher $\mathcal{B}$, which must distinguish with negligible probability only.
- Unconditional security considers a setting where the security does not depend on "what is only assumed but not known to be true". For example, if the RSA problem were *known* to be hard for PPT adversaries (in contrast to merely *assuming* the hardness), then a commitment scheme using the RSA problem would be considered unconditionally secure for PPT adversaries. From our point of view, unconditional security rules out the use of trusted setups.

2.3. Cryptographic Building Blocks

Starting with basic cryptographic primitives, we present definitions of important cryptographic building blocks that are used throughout this thesis.

2.3.1. One-Way Functions and Trapdoor Permutations

One-way functions constitute the perhaps most basic cryptographic primitive. While we will subsequently often use stronger building blocks, their definition is important starting point.

Definition 2.4 (One-Way Function, adapted from [G01])**.** A function $f : \{0,1\}^* \to \{0,1\}^*$ is called *one-way* if the following conditions are satisfied:

1. There exists a (deterministic) polynomial-time algorithm F such that for every $x \in \{0,1\}^*$, it holds that $f(x) = F(x)$.
2. For every PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for every $\kappa \in \mathbb{N}$ and every $z \in \{0,1\}^*$, it holds that

$$\Pr[\mathcal{A}(1^\kappa, z, f(x)) \in f^{-1}(f(x)) \mid x \overset{\$}{\leftarrow} \{0,1\}^\kappa] \leq \mathsf{negl}(\kappa)$$

   where the probability is over the choice of x and the coins of $\mathcal{A}$.

The definition of one-way functions can easily be adapted to one-way permutations. If there additionally exists a secret trapdoor that allows to efficiently invert the permutation, we obtain trapdoor permutations.

Definition 2.5 (Trapdoor Permutation, based on [G04])**.** A *collection of trapdoor permutations* is a collection of permutations $\{f_\alpha\}_\alpha$ accompanied by four PPT algorithms (I, S, F, B) such that the following conditions hold:

1. On input 1^κ, algorithm I selects a random κ-bit long index α of a permutation f_α, along with a corresponding trapdoor τ.
2. On input α, algorithm S samples the domain of f_α, returning an almost uniformly distributed element in it.
3. For x in the domain of f_α, given α and x, algorithm F returns $f_\alpha(x)$ (*i.e.* $F(\alpha, x) = f_\alpha(x)$).
4. For y in the range of f_α if (α, τ) is the possible output of $I(1^\kappa)$, then, given τ and y, algorithm B returns $f_\alpha^{-1}(y)$ (*i.e.* $B(\tau, y) = f_\alpha^{-1}(y)$).
5. For every PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for every $\kappa \in \mathbb{N}$ and every $z \in \{0,1\}^*$,

   $$\Pr[\mathcal{A}(1^\kappa, z, \alpha, y) = f_\alpha^{-1}(y) \mid (\alpha, \tau) \leftarrow I(1^\kappa), y \leftarrow S(\alpha)] \leq \mathsf{negl}(\kappa)$$

   where the probability is over the coins of $\mathcal{A}$, I and S.

Often, it may be desirable to relax the first condition such that the index or *public description* α may be of polynomial length in the security parameter.

For some of our applications, we need trapdoor permutations with an additional property: Even given the coins used by the sampling algorithm S, it must be hard to invert the permutation. Trapdoor permutations satisfying this property are called *enhanced* trapdoor permutations and defined as follows.

Definition 2.6 (Enhanced Trapdoor Permutation, [G04])**.** A collection of trapdoor permutations is called *enhanced* if for every PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for every $\kappa \in \mathbb{N}$ and every $z \in \{0,1\}^*$,

$$\Pr[\mathcal{A}(1^\kappa, z, \alpha, r) = f_\alpha^{-1}(y) \mid (\alpha, \tau) \leftarrow I(1^\kappa), r \overset{\$}{\leftarrow} \{0,1\}^{\mathsf{poly}(\kappa)}, y = S(\alpha; r)] \leq \mathsf{negl}(\kappa)$$

where the probability is over the coins of $\mathcal{A}$, I and the choice of r.

We also consider trapdoor permutations with *dense public description*, implicitly introduced in [DP92]. Informally, the dense public description property guarantees that a random string α of appropriate length is computationally indistinguishable from public descriptions output by I.

Definition 2.7 (Trapdoor Permutation with Dense Public Description)**.** Let $l : \mathbb{N} \to \mathbb{N}$ be an efficiently computable algorithm that, given the security parameter as input, outputs the length of public parameters α generated with this security parameter. A collection of trapdoor permutations has a *dense public description* if the ensembles $\{\alpha \mid (\alpha, \tau) \leftarrow I(1^\kappa)\}_{\kappa \in \mathbb{N}}$ and $\{\alpha' \mid \alpha' \overset{\$}{\leftarrow} \{0,1\}^{l(\kappa)}\}_{\kappa \in \mathbb{N}}$ are computationally indistinguishable.

Due to the above condition, the (enhanced) hardness also holds for public descriptions that are sampled uniformly at random. In particular, trapdoor permutations with dense public description allow their public description to be sampled via a coin-toss.

2.3.2. Commitment Schemes

We start with giving an overview of *commitment schemes*, which are two-party protocols between a committer and a receiver. In a first phase, the committer C commits to a secret value, which it can unveil in the subsequent unveil phase to the receiver R.

Commitment schemes exist in various variants and with various properties. Here, we will only give a short overview and define the most basic properties. These definitions will be extended and adapted as needed in the following chapters.

Definition 2.8 (Interactive Commitment Scheme)**.** An *interactive* commitment scheme $\mathsf{COM} = \langle \mathsf{C}, \mathsf{R} \rangle$ with message space M is a two-phase protocol between two interactive Turing machines C and R. Both parties get the phase `commit` or `unveil` as well as the security parameter κ as common input. The committer C additionally gets a value $v \in M$ as input for the commit phase. We call the transcript

$$c \leftarrow \langle \mathsf{C}(v), \mathsf{R}(\varepsilon) \rangle (1^\kappa, \texttt{commit})$$

the commitment c between C and R (to $v \in M$).

We say that COM is perfectly correct if for every $\kappa \in \mathbb{N}$ and every $v \in M$,

$$\begin{aligned}\Pr[v' = v \mid (\mathit{view}_{\mathsf{C}}, \mathit{view}_{\mathsf{R}}, c) \leftarrow \operatorname{out}\langle \mathsf{C}(v), \mathsf{R}(\varepsilon) \rangle (1^\kappa, \texttt{commit}),\\ v' \leftarrow \operatorname{out}_{\mathsf{R}}\langle \mathsf{C}(\mathit{view}_{\mathsf{C}}), \mathsf{R}(\mathit{view}_{\mathsf{R}}) \rangle (\texttt{unveil})] = 1\end{aligned}$$

where the probability is over the coins used by C and R. Statistical correctness is defined analogously and admits a negligible error probability.

Definition 2.8 does not guarantee any security properties such as the *hiding* or the *binding* property yet. Informally, the hiding property protects the committer by guaranteeing that a (malicious) receiver does not learn anything about the value committed to until the commitment is opened.

Definition 2.9 (Hiding)**.** For an interactive commitment scheme $\mathsf{COM} = \langle \mathsf{C}, \mathsf{R} \rangle$, the random variable $\mathrm{Exp}^{\mathrm{Hiding}}_{\mathcal{A},\mathsf{COM}}(\kappa, z)$ is defined as follows:

1. Run $\mathcal{A}$ on input $(1^\kappa, \texttt{find}, z)$ and obtain $(m_0, m_1, \mathit{state})$.
2. Sample a uniformly random bit $b \xleftarrow{\$} \{0, 1\}$.
3. If $|m_0| \neq |m_1|$, return b.
4. Otherwise, obtain $b' \leftarrow \operatorname{out}_{\mathcal{A}}\langle \mathsf{C}(m_b), \mathcal{A}(\texttt{guess}, \mathit{state}) \rangle (1^\kappa, \texttt{commit})$.
5. If $b = b'$, output 1. Otherwise, output 0.

An adversary $\mathcal{A}$ is called *valid* if $m_0, m_1 \in M$ and $\mathcal{A}$ eventually outputs a single bit.

Let $\mathrm{Adv}^{\mathrm{Hiding}}_{\mathcal{A},\mathsf{COM}}(\kappa, z) = |\Pr[\mathrm{Exp}^{\mathrm{Hiding}}_{\mathcal{A},\mathsf{COM}}(\kappa, z) = 1] - \frac{1}{2}|$ denote the advantage of a (valid) adversary $\mathcal{A}$. The probability is over the coins of $\mathcal{A}$, C and the choice bit b.

We say that COM is computationally hiding if for every valid PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for every $\kappa \in \mathbb{N}$ and every $z \in \{0,1\}^{\kappa}$, it holds that $\mathrm{Adv}^{\mathrm{Hiding}}_{\mathcal{A},\mathsf{COM}}(\kappa, z) \leq \mathsf{negl}(\kappa)$. If the advantage is also negligible for every unbounded adversary, then we say that COM is statistically hiding. If the advantage is only negligible for adversaries that perform a bounded number of steps $\ell(\kappa)$, we say that COM is (timed) $\ell(\kappa)$-hiding.

Conversely, the binding property protects an honest receiver from a (malicious) committer by ensuring that the committer can open the commitment to a single value only.

Definition 2.10 (Binding)**.** For an interactive commitment scheme $\mathsf{COM} = \langle \mathsf{C}, \mathsf{R} \rangle$, the random variable $\mathrm{Exp}^{\mathrm{Binding}}_{\mathcal{A},\mathsf{COM}}(\kappa, z)$ is defined as follows:

1. Perform the commit phase and save the state of $\mathcal{A}$ as $view_{\mathcal{A}}$, the state of R as $view_{\mathsf{R}}$ and the commitment as c, *i.e.* $(view_{\mathcal{A}}, view_{\mathsf{R}}, c) \leftarrow \mathrm{out}\langle \mathcal{A}(z), \mathsf{R}(\varepsilon) \rangle (1^{\kappa}, \mathtt{commit})$.
2. Perform the unveil phase on the previously saved states and obtain the value m_0 accepted by the receiver, *i.e.* $m_0 \leftarrow \mathrm{out}_{\mathsf{R}}\langle \mathcal{A}(view_{\mathcal{A}}, 0), \mathsf{R}(view_{\mathsf{R}}) \rangle (\mathtt{unveil})$.
3. Perform the unveil phase again on the states from the commit phase and obtain the value m_1 accepted by the receiver, *i.e.* $m_1 \leftarrow \mathrm{out}_{\mathsf{R}}\langle \mathcal{A}(view_{\mathcal{A}}, 1), \mathsf{R}(view_{\mathsf{R}}) \rangle (\mathtt{unveil})$.
4. Return 1 if the receiver accepted different valid values, *i.e.* if $m_0 \in M \land m_1 \in M \land m_0 \neq m_1$ and return 0 otherwise.

$\mathcal{A}$ and R use the same coins in each run of the unveil phase.

Let $\mathrm{Adv}^{\mathrm{Binding}}_{\mathcal{A},\mathsf{COM}}(\kappa, z) = \Pr[\mathrm{Exp}^{\mathrm{Binding}}_{\mathcal{A},\mathsf{COM}}(\kappa, z) = 1]$ denote the advantage of a possibly malicious committer $\mathcal{A}$. The probability is over the coins of $\mathcal{A}$ and R.

We say that COM is computationally binding if for every PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for every $\kappa \in \mathbb{N}$ and every $z \in \{0,1\}^*$, $\mathrm{Adv}^{\mathrm{Binding}}_{\mathcal{A},\mathsf{COM}}(\kappa, z) \leq \mathsf{negl}(\kappa)$. We say that COM is statistically binding if for every unbounded adversary $\mathcal{A}$, it holds that $\mathrm{Adv}^{\mathrm{Binding}}_{\mathcal{A},\mathsf{COM}}(\kappa, z) \leq \mathsf{negl}(\kappa)$. If the advantage is zero for unbounded adversaries, we say that COM is perfectly binding.

If COM is statistically binding, then a commitment c statistically defines the unique value v committed to such that $v \in M$ or $v = \bot$ if the commitment is invalid, except with negligible probability.

2.4. Universal Composability and its Variants

In the following, we give a very brief introduction of the Universal Composability framework due to Canetti [C01] and the corresponding security notion called UC security. It focuses only on the aspects pertaining to this thesis, omitting other aspects and details.

The Setting. Extending the notion of (stand-alone) real-ideal security, which considers a single protocol execution and non-reactive functionalities only, Universal Composability models the distributed execution of protocols in a network environment by multiple parties. Like in the real-ideal paradigm, the security of protocols is defined through *ideal functionalities* that model a trusted entity carrying out some task honestly, *e.g.* commitments, authenticated message transmission or secure function evaluation. UC security considers the security of a protocol not only when a single instance is executed in isolation, but in a setting where multiple, possibly different and adversarially chosen, protocols are executed concurrently.

The Machine Model. Protocol parties and other entities of the framework are modeled as *interactive Turing machines* that feature several communication tapes. *Instances of interactive Turing machines (ITIs)* have a party ID (PID) and a session ID (SID) that comprise, together with their code, their *extended identity*, which is unique throughout an execution. An ITI may receive input via its *input tape*, receive computation results of subroutines via its *subroutine output tape* and receive general messages (from the adversary) via its *backdoor tape*. Sending a message is performed by writing the message, along with the recipient's extended identity and the destination tape, to the *outgoing message tape* and issuing a special `external write` instruction. Messages are not delivered directly, but governed by a so-called *control function*, which may allow or disallow the `external write` request as well as adjust certain parameters (based on the rules of the execution experiment). All entities in the UC framework run in (non-uniform) probabilistic polynomial time[1].

The Execution Experiment. The UC execution is, apart from the challenge protocol and its parties, defined using two further entities: An adversary that may corrupt parties and, depending on the communication model, alter, inject or drop messages sent between protocol parties via the backdoor tape. The adversary may freely interact with the environment $\mathcal{Z}$ that is the first machine to be invoked and (adaptively) provides input to all protocol parties and the adversary. In the end, the environment outputs a single bit.

As environment and adversary may freely communicate throughout the execution, the environment is able to incorporate other protocols executed concurrently next to the challenge protocol, capturing the setting of general concurrent composition. In order to show the UC security of a protocol, it is thus not necessary to explicitly consider different protocols running alongside as they can, without loss of generality, be considered to be part of the environment.

[1]As the exact definition of polynomial time within Universal Composability (UC) is of no concern for the results of this thesis, we do not provide a definition and refer the interested reader to [C20].

Ideal Functionalities and Protocols In contrast to stand-alone real-ideal security, UC security does not distinguish between a real and an ideal execution experiment but provides a uniform treatment of protocol emulation. To this end, there exists a special class of protocols called *ideal protocols*. For an ideal functionality $\mathcal{F}$, let $\mathrm{IDEAL}(\mathcal{F})$ denote the corresponding ideal protocol. Informally, an ideal protocol consists of *dummy parties* that have no functionality besides forwarding input to their ideal functionality and, vice versa, forwarding output. However, they are necessary in order to establish the proper interface to an ideal functionality.

Protocol Emulation. UC security is based on the notion of protocol emulation: Let π and ϕ be PPT protocols, *i.e.* descriptions of PPT interactive Turing machines that interact with each other. We say that π *(UC-)emulates* ϕ (denoted by $\pi \geq_{\mathrm{UC}} \phi$) if, informally, for every PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that no PPT environment $\mathcal{Z}$ can distinguish if it interacts with π and $\mathcal{A}$ or with ϕ and $\mathcal{S}$. If $\pi \geq_{\mathrm{UC}} \mathrm{IDEAL}(\mathcal{F})$, we say that π *(UC-)realizes* $\mathcal{F}$.

Statistical Protocol Emulation. The notion of UC protocol emulation can be adapted to *statistical* emulation: We say that a PPT protocol π *statistically UC-emulates* a PPT protocol ϕ if for every adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ with runtime polynomial in the runtime of $\mathcal{A}$ such that for every environment $\mathcal{Z}$, the output of $\mathcal{Z}$ in the execution with π and $\mathcal{A}$ is statistically indistinguishable from the output of $\mathcal{Z}$ in the execution with ϕ and $\mathcal{S}$. If π statistically UC-emulates ϕ, we write $\pi \geq_{\text{Stat-UC}} \phi$.

Properties of UC Security. UC security exhibits several properties such as reflexivity or transitivity and is furthermore closed under general concurrent (*i.e.* universal) composition for the class of *subroutine-respecting* PPT protocols. Very informally, a protocol is *subroutine-respecting* if all communication with "outside" the protocol only happens through a protocol's *main parties*, *i.e.* the parties specified in the protocol but not sub-parties invoked by these main parties, providing a well-defined interface. Moreover, the subroutine-respecting property guarantees that machines are only subroutine of *one* protocol session, establishing intra-protocol boundaries that allow well-defined protocol replacement in the first place.

Using these properties, one can show that one instance of π UC-emulates a single instance of the protocol ϕ to argue that one may replace *all* top-level sessions (*i.e.* sessions of ϕ that are not invoked by main parties of a session of ϕ) of ϕ in a protocol ρ that makes multiple subroutine calls to ϕ (we say that ρ is in the ϕ-hybrid model) with sessions of π without losing security, *i.e.* $\rho^{\phi\to\pi} \geq_{\mathrm{UC}} \rho$. Here, $\rho^{\phi\to\pi}$ is the protocol ρ where sessions of ϕ are replaced with sessions of π.

Often, this composition theorem is used together with the transitivity property to conclude that for a protocol ρ in the $\mathcal{F}$-hybrid model that UC-realizes $\mathcal{G}$, one can replace $\mathcal{F}$ by its realization π and the resulting protocol $\rho^{\mathrm{IDEAL}(\mathcal{F})\to\pi}$ still UC-realizes $\mathcal{G}$.

As the dummy adversary that reports all messages between protocol parties to the environment and delivers all messages sent from the environment is complete, it is sufficient to show UC security only for this dummy adversary, leading to easier proofs.

Impossibility Results. While UC security is a very strong notion, there exist a number of impossibility results such as the ones of [CF01; CKL03; PR08; KL11]. that imply that non-trivial functionalities such as the commitment functionality $\mathcal{F}_{\mathrm{COM}}$ can, under many natural circumstances, only be realized using some kind of setup, *e.g.* a common reference string, a public key infrastructure or a random oracle. Lindell [L03] has shown that these impossibilities are not due to the particular definition of UC security, but universally apply to similar settings.

2.4.1. Variants of UC Security

Generalized Universal Composability. A major drawback of UC security is that the composition theorem only holds for *subroutine-respecting* protocols. As a consequence, ideal functionalities are always local to *one* protocol instance. Not only may this exacerbate the problem of not having an entity that is trusted enough to provide the necessary setups, but arguably does not model the reality where setups such as public key infrastructures or smart cards are used for multiple protocols. Generalized Universal Composability (GUC) security [CDPW07] extends UC security to a setting where *global* ideal functionalities exist, which may be used by multiple protocols. Like UC security, GUC security features universal composability for protocols that are subroutine-respecting *except* with respect to the global ideal functionality $\mathcal{G}$. Such protocols are called $\mathcal{G}$-subroutine-respecting. However, it is subject to even stronger impossibility results [CDPW07]. Recently, definitional imprecisions with respect to GUC security have been identified and subsequently addressed using a new formalism [BCH$^+$20]. As these problems are not of particular relevance in our setting, we will stick to GUC security where applicable.

SPS and ($\mathcal{C}_{\mathsf{env}}, \mathcal{C}_{\mathsf{sim}}$) Security. The first notion for composable secure multi-party computation (MPC) in the plain model is called security with super-polynomial simulation (SPS) and was introduced by Pass [P03]. With SPS security, the simulator may run in super-polynomial time, circumventing the known impossibility results of UC security. Due to the simulator being more powerful than the environment, SPS security features no universal composability. Furthermore, the reusability of UC-secure protocols is limited, as a protocol's security may be affected by the powerful simulator. The notion of SPS security has subsequently been generalized to ($\mathcal{C}_{\mathsf{env}}, \mathcal{C}_{\mathsf{sim}}$) security [LPV09]. Here, $\mathcal{C}_{\mathsf{env}}$ and $\mathcal{C}_{\mathsf{sim}}$ denote the complexity classes of the environment and the simulator.

Universal Composability with Super-polynomial Helpers. Defined within the framework of GUC security, *UC with super-polynomial helpers* [CLP10] considers a special global functionality called *helper*. In contrast to ordinary ideal functionalities, which are usually PPT, the helper of [CLP10] is able to perform inefficient (*i.e.* super-polynomial-time) computations, allowing to extract statistically binding commitments using brute force. In this setting, the impossibility results of UC security no longer apply and composable MPC in the plain model can be achieved. A conceptually similar approach was proposed by Prabhakaran and Sahai [PS04], where an inefficient *"Imaginary Angel"* was used instead of a helper.

Long-Term Security. Long-term universal composability [MU10] extends the notion of universal composability to a setting where *all* hardness assumptions eventually become invalid. 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 PPT protocol π long-term-emulates a PPT protocol ϕ if for every PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for every PPT environment $\mathcal{Z}$, 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. If π long-term-emulates ϕ, we write $\pi \geq_{\text{LT-UC}} \phi$.

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 2.11 (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 $\mathit{trans} \setminus \mu$ 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 $\mathit{trans} = f(\kappa, \mathit{trans} \setminus \mu)$, where $\kappa \in \mathbb{N}$ 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 2.1. 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 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 2.1 (Examples for Long-Term Revealing Functionalities, adapted from [MU10]). *Coin toss ($\mathcal{F}_{\mathrm{CT}}$) and CRS ($\mathcal{F}_{\mathrm{CRS}}$) with any distribution D are LTR for all parties. Commitment ($\mathcal{F}_{\mathrm{COM}}$) and ZK ($\mathcal{F}_{\mathrm{ZK}}$) 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 $\mathcal{F}_{\mathrm{KRK}}$ parameterized with G is LTR for all parties registered with $\mathcal{F}_{\mathrm{KRK}}$.*

Functionalities that are long-term revealing for the committer (*e.g.* the ones in Lemma 2.1) cannot be used to long-term-UC-realize the ideal functionality for commitments $\mathcal{F}_{\mathrm{COM}}$ [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 $\mathcal{F}_{\mathrm{CRS}}$-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 Remark 4.33.

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 [MMSU22] 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.

2.4.2. Ideal Functionalities

As UC security (usually) considers a setting where message delivery is adversarially controlled, it is desirable to allow the adversary suppress outputs to parties even in the execution of the ideal protocol associated with an ideal functionality. A convenient way to model this is through the mechanism of *delayed outputs*: If an ideal functionality generates a delayed output to some party P, the adversary is first notified of this output. It can then, at any time, notify the functionality that it allows the output. Only then, P receives the output from the ideal functionality. We distinguish between *public* and *private* delayed outputs. In the former case, the adversary learns the output's value, whereas in the latter case, the adversary may only learn some "public header" or even nothing at all, except for the fact the functionality wants to generate an output to P.

We now present important ideal functionalities that are used throughout this thesis, starting with the ideal functionality $\mathcal{F}_{\mathrm{CRS}}$ models a common reference string that is accessible by all parties.

Functionality $\mathcal{F}_{\mathrm{CRS}}$

$\mathcal{F}_{\mathrm{CRS}}$ proceeds as follows, parameterized with a security parameter κ, a family of efficiently samplable distributions $\{D_\kappa\}_{\kappa \in \mathbb{N}}$ and an adversary $\mathcal{S}$.

1. When activated for the first time on input $(\texttt{value}, sid)$ by a party P, sample a value $d \leftarrow D_\kappa$ and generate a public delayed output $(\texttt{value}, sid, d)$ to P. Answer subsequent $(\texttt{value}, sid)$ inputs from parties P' by generating a public delayed output $(\texttt{value}, sid, d)$ to P'.

Figure 2.1.: The ideal functionality for a common reference string $\mathcal{F}_{\mathrm{CRS}}$, adapted from [CF01].

T ideal functionality for commitments $\mathcal{F}_{\mathrm{COM}}$ allows a party C to commit to a bit b and to unveil it later on.

Functionality $\mathcal{F}_{\mathrm{COM}}$

$\mathcal{F}_{\mathrm{COM}}$ proceeds as follows, parameterized with a security parameter κ, running with a committer C and a receiver R and an adversary $\mathcal{S}$.

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

2. Upon receiving an input $(\texttt{unveil}, sid)$ from C, generate a public delayed output $(\texttt{unveil}, sid, b)$ to R if there has been a previously delivered `committed` output. Otherwise, ignore this input. Ignore subsequent `unveil` inputs.

Figure 2.2.: The ideal functionality for commitments $\mathcal{F}_{\mathrm{COM}}$, adapted from [CF01].

Often, it may be helpful to consider a single ideal functionality that captures multiple bilateral commitments between several parties instead of several instances of $\mathcal{F}_{\mathrm{COM}}$. In particular, this may be the case if i) the same setup should be used for all commitments or ii) if there is no universal composition theorem and it is thus beneficial to directly show that a protocol π can be used for multiple commitments concurrently, *i.e.* is concurrently self-composable. The ideal functionality $\mathcal{F}_{\mathrm{MCOM}}$ in Figure 2.3, introduced by [CF01], models this. Individual commitments are distinguished by their commitment ID *cid*.

Functionality $\mathcal{F}_{\text{MCOM}}$

$\mathcal{F}_{\text{MCOM}}$ proceeds as follows, parameterized with a security parameter κ, running with parties $P_1, \dots, P_n$ and an adversary $\mathcal{S}$.

1. Upon receiving an input $(\texttt{commit}, \mathit{sid}, \mathit{cid}, P_i, P_j, b)$ from P_i, where $b \in \{0,1\}$, record the tuple $(\mathit{sid}, \mathit{cid}, P_i, P_j, b)$ and generate a public delayed output $(\texttt{committed}, \mathit{sid}, \mathit{cid}, P_i, P_j)$ to P_j. Ignore subsequent $(\texttt{commit}, \mathit{sid}, \mathit{cid}, P_i, P_j, \star)$ inputs.
2. Upon receiving an input $(\texttt{unveil}, \mathit{sid}, \mathit{cid}, P_i, P_j)$ from P_i, proceed as follows: If a tuple $(\mathit{sid}, \mathit{cid}, P_i, P_j, b)$ is recorded, generate a public delayed output $(\texttt{unveil}, \mathit{sid}, \mathit{cid}, P_i, P_j, b)$ to P_j. Otherwise, do nothing. Ignore subsequent $(\texttt{unveil}, \mathit{sid}, \mathit{cid}, P_i, P_j)$ inputs.

Figure 2.3.: The ideal functionality for multiple commitments $\mathcal{F}_{\text{MCOM}}$, adapted from [CF01].

The ideal functionality for zero-knowledge $\mathcal{F}_{\text{ZK}}$ allows a prover P to prove the validity of a statement x to a verifier V relative to a relation R, ensuring that the prover knows an appropriate witness.

Functionality $\mathcal{F}_{\text{ZK}}$

$\mathcal{F}_{\text{ZK}}$ proceeds as follows, parameterized with a security parameter κ, a relation R, running with a prover P, a verifier V and an adversary $\mathcal{S}$.

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

Figure 2.4.: The ideal functionality for zero-knowledge $\mathcal{F}_{\text{ZK}}$, adapted from [CLOS02].

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

We finish this section with the following terminology.

Functionality $\mathcal{F}_{\mathrm{CP}}$

$\mathcal{F}_{\mathrm{CP}}$ proceeds as follows, parameterized with a security parameter κ, a relation R, running with a committer C, a receiver V and an adversary $\mathcal{S}$.

1. Commit phase: Upon receiving an input $(\texttt{commit}, \mathit{sid}, w)$ from C where $w \in \{0,1\}^{\kappa}$, append the value w to the list $\overline{w}$ and generate a public delayed output $(\texttt{receipt}, \mathit{sid})$ to V. (Initially, the list $\overline{w}$ is empty.)

2. Prove phase: Upon receiving an input $(\texttt{CP-prover}, \mathit{sid}, x)$ from C, where $x \in \{0,1\}^{\mathsf{poly}(\kappa)}$, compute $R(x, \overline{w})$: If $R(x, \overline{w}) = 1$, generate a public delayed output $(\texttt{CP-proof}, \mathit{sid}, x)$ to V. Otherwise, ignore the input.

Figure 2.5.: The ideal functionality for commit-and-prove $\mathcal{F}_{\mathrm{CP}}$, adapted from [CLOS02].

Terminology 1 (Constant-Round Realization)**.** Let $\mathcal{F}$ be an ideal functionality. Let π be a protocol such that $\pi \geq \mathrm{IDEAL}(\mathcal{F})$, where $\geq$ denotes some notion of protocol emulation. We say that π is a *constant-round realization of* $\mathcal{F}$ if for every round of $\mathcal{F}$, π has $O(n)$ rounds, where n is the number of main parties of π resp. $\mathcal{F}$.

The above terminology is motivated by the fact that realizing $\mathcal{F}$ may *inherently* require a super-constant number of rounds (*e.g.* because $\mathcal{F}$ is reactive and accepts an unbounded number of inputs), even if the per-round and per-party round complexity of the protocol π realizing $\mathcal{F}$ may be constant. In particular, it is in line with the notion of constant-round protocols used in *e.g.* [BDH+17].

2.5. Environmental Friendliness

UC security has the desirable property of *environmental friendliness* [CLP13a], which, informally, ensures that game-based security properties of protocols running along UC protocols ("in the environment") are not impacted by the UC execution. Unfortunately, this property does not hold for *all* game-based security properties for many notions that allow composable MPC in the plain model due to the use of super-polynomial simulation. What is more, determining whether the game-based property holds may be non-trivial, requiring *e.g.* to consider the security proof of the protocol in question [CLP13a].

Environmental friendliness considers the validity of a game-based security property P when the presumptively P-secure cryptographic scheme Π, where P is defined via some security game, is executed alongside another protocol Π' with game-based security property Q. In particular, a UC-like execution running concurrently is considered.

First, we restate the notion of a *security game* as defined in [CLP13a].

Definition 2.12 (Security Game, [CLP13a])**.** A *security game* (or game) consists of an interactive Turing machine (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 $\tau_{\mathcal{C}} + \varepsilon$. 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 indistinguishability under chosen plaintext attack (IND-CPA) game for encryption schemes with $\tau = 1/2$, *i.e.* the trivial winning probability of an adversary. However, Definition 2.12 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 2.13 (Game-Based Assumptions, [CLP13a])**.** A *game-based assumption* is simply a security game $\mathcal{C} = (\mathsf{Chal}, \tau)$, 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 every $n \in \mathbb{N}$. We say that assumption $\mathcal{C}$ holds if no non-uniform PPT adversary can break the game (Chal, τ).

Definition 2.13 rules out $\tau = 1$ for IND-CPA security, as there is no trivial winning strategy for the IND-CPA game with winning probability 1 (if the encryption scheme under analysis is indeed IND-CPA-secure). 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 2.14 (Game-Based Security Property, [CLP13a])**.** A *game-based security property* of a cryptographic scheme Π is simply a security game $P_\Pi = (\mathsf{Chal}, \tau)$. 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 [CLP13a] gives an informal description of the associated game (see the full version [CLP13b] 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 $\mathsf{Chal}^{\mathcal{G}/\rho}$ 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 2.15 (Environmental Friendliness, adapted from [CLP13a]). Let $P = (\mathsf{Chal}, \tau)$ 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 *environmentally friendly* to Π with property P, if the security property $P^{\mathcal{G}/\rho} = (\mathsf{Chal}^{\mathcal{G}/\rho}, \tau)$ holds.

3. Environmentally Friendly Composable Multi-Party Computation in the Plain Model from Standard (Timed) Assumptions

This chapter is largely based on a major revision of the full version of the paper [BMM21] (© IACR 2021, DOI 10.1007/978-3-030-90459-3_25).

Abstract

Starting with the work of Rivest *et al.* in 1996, timed assumptions have found many applications in cryptography, building *e.g.* the foundation of the blockchain technology. They also have been used in the context of classical secure multi-party computation (MPC), *e.g.* to enable fairness. We follow this line of research to obtain composable general MPC in the plain model.

This approach comes with a major advantage regarding *environmental friendliness*, a property coined by Canetti, Lin and Pass (FOCS 2013). Informally, this means that our constructions do not "hurt" game-based security properties of protocols that hold against polynomial-time adversaries when executed alone.

As an additional property, our constructions can be plugged into *any* UC-secure protocol without loss of security, which is not possible with previous approaches for composable MPC in the plain model.

Towards proving the security of our constructions, we introduce a variant of the UC security notion that allows the use of building blocks with timed security properties, for example timed commitment schemes. Combining standard timed assumptions and standard polynomial-time hardness assumptions, we construct a composable commitment scheme in the plain model. This composable commitment scheme can then be plugged into a UC-secure general MPC protocol in the commitment-hybrid model.

This way, we obtain the *first fully* environmentally friendly composable constant-round black-box general MPC protocol in the plain model from standard (timed) assumptions.

3.1. Introduction

In order to achieve the very strong notion of Universal Composability (UC) [C01], trusted setups are required [CF01]. However, in practice, trusted setups are often hard to come by. Therefore, a long line of research (*e.g.* [P03; BS05; LPV09; GGJS12; GKP18;

DMRV13; PS04; CLP10; CLP13a; BDH$^+$17]) has investigated how composable MPC can be obtained in the plain model, *i.e.* only assuming authenticated communication (and possibly cryptographic hardness assumptions).

Common to their techniques is that the simulation is *environmentally unfriendly*, *i.e.* "hurts" the security of protocols that run along-side and that rely on polynomial-time hardness assumptions.

Formally, this is captured by the notion of *environmental friendliness* as defined by Canetti, Lin, and Pass [CLP13a], which considers all game-based security properties of a protocol against polynomial-time adversaries.

The typical reason for limited environmental friendliness is a super-polynomial simulation (SPS), which can break polynomial-time assumptions used in other protocols, therefore impacting their security properties. This holds even if the super-polynomial resources are restricted by *e.g.* an "Imaginary Angel" or a super-polynomial helper.

However, SPS techniques are not the only danger to the security of other protocols: Non-uniform advice given to the simulator (*e.g.* as in [LPV09]) may impact the security of previously started protocols—even if they are concurrently composable and secure against non-uniform adversaries. This additional property is not considered by the definition of environmental friendliness.

Ever since composable MPC in the plain model has been investigated, the following question has been left unanswered:

Can we achieve composable MPC in the plain model that is friendly to protocols that are executed along-side and may have started previously?

Previous results suggest that a simulation technique that runs in polynomial time and does not rely on non-uniform advice is needed. Such a simulation cannot be achieved, in principle, even by previous advanced approaches like Angel-based security [PS04; CLP10; CLP13a] or shielded super-polynomial simulation [BDH$^+$17]. Therefore, new techniques to overcome the impossibility results of UC security are needed.

With the advent of the blockchain era, timed cryptographic assumptions, *i.e.* assumptions that are only assumed to hold against adversaries with a certain runtime bound, have seen widespread use in the real world. A very popular example is the *proof of work* protocol of the Bitcoin blockchain. Even though its hardness is not based on some well-understood cryptographic assumption, it has proven to work nevertheless for many years.

Timed variants of classic cryptographic primitives such as commitment schemes can be constructed from timed assumptions that are inspired by well-understood standard assumptions. Rivest, Shamir, and Wagner [RSW96] have initiated this study and proposed a *time-lock puzzle* based on the hardness of factoring and the time required to square modulo a composite. Based on such assumptions, timed cryptographic primitives such as time-lock puzzles and timed-release encryption [RSW96] or timed signatures and timed commitment schemes [BN00] can be constructed in the plain model. More recently, stronger primitives such as non-malleable time-lock puzzles and commitment schemes have been constructed [EFKP20; KLX20] using a setup.

As timed assumptions (and primitives) often can be broken in polynomial time by definition, they seem destined to solve the problem of limited friendliness exhibited by previous approaches for composable MPC in the plain model. In the following, we thus investigate the following questions:

Can we use timed assumptions to achieve composable MPC in the plain model? What are the advantages and disadvantages of such an approach?

We answer the first question affirmatively and propose a new approach for general MPC in the plain model based on asymmetries that are only temporary and much smaller compared to previous approaches. Namely, these asymmetries consist of only a polynomial number of computation steps sufficient to leverage timed cryptographic assumptions. The very feasibility of this approach may seem surprising as timed cryptographic primitives eventually lose (some or all) security guarantees. For example, timed commitments will eventually leak their secret by definition. Previous constructions crucially rely on this not to happen, *i.e.* the complexity asymmetry and the ensuing hardness to hold throughout the whole execution and beyond. We side-step this problem by using timed primitives to merely set up short-lived trapdoors that can only be used while the primitives are secure. After their security has expired, the (now possibly leaked) trapdoor is useless for the adversary. Yet, a simulator can use it to establish a long-lived trapdoor based on some classical polynomial-time assumption.

We introduce the notion of *Time-Lock UC (TLUC)* security, which is based on UC security and cast in the unmodified UC framework. With TLUC, honest parties may set up timers with some timeout $\ell \in \mathbb{N}$ that expire when all entities have spent more than ℓ steps in total. This allows to use (stand-alone) timed primitives such as time-lock puzzles or timed commitment schemes within larger protocols. While computations performed by protocol parties, environment and adversary are counted against timers, computations performed by the simulator are not. This allows simulators to break timed assumptions "at no cost" in terms of time accounting, while remaining polynomially bounded. Such a simulator can then, for example, extract a timed commitment while it is still hiding for environment and adversary.

With respect to the question of environmental friendliness, it suffices to see that the notion of TLUC security is a meaningful special case of UC security, which is *environmentally friendly* according to the definition of [CLP13a]. This already implies that our notion features the same environmental friendliness.

In order to be friendly to previously started protocols, a uniform simulation, *i.e.* one that does not rely on non-uniform advice, is needed. Looking ahead, this is indeed achieved by our constructions.

To the best of our knowledge, we are the first to achieve both of these properties simultaneously.

Leveraging timed assumptions for composability comes with a number of additional advantages. Namely, our notion is UC-compatible in the sense that if π UC-emulates ϕ for arbitrary PPT protocols π and ϕ, then π also TLUC-emulates ϕ. TLUC security allows the reuse of UC protocols in the sense that one can take a UC-secure protocol ρ

making one subroutine call to $\mathcal{F}$ that UC-realizes some ideal functionality $\mathcal{G}$ and replace $\mathcal{F}$ with its TLUC realization π. The composite protocol $\rho^{\mathcal{F}\to\pi}$ is then guaranteed to TLUC-realize $\mathcal{G}$. These properties are not generally offered in full by other notions that allow composable general MPC in the plain model and are not implied by (limited) environmental friendliness. What is more, TLUC security is meaningful for ideal functionalities that rely on (even uniform) polynomial-time assumptions. This is in contrast to *e.g.* SPS security, where such functionalities are affected by the super-polynomial simulator or non-uniform simulation [LPV09].

Unfortunately, TLUC security is not closed under composition for protocols using timers. Thus, one has to manually prove that multiple instances of π TLUC-realize multiple instances of $\mathcal{F}$ (*i.e.* $\hat{\pi}$ TLUC-realizes $\widehat{\mathcal{F}}$).

Like previous approaches for general MPC in the plain model and even UC security, TLUC security may not be friendly to *timed* game-based properties of other protocols, *e.g.* the timed hiding property of a timed commitment scheme. This property is neither captured by the definition of environmental friendliness nor fulfilled by any previous notion that allows composable MPC—not even UC security.

Towards realizing composable general MPC, we first construct a commitment scheme that TLUC-realizes the ideal functionality for multiple commitments $\mathcal{F}_{\text{MCOM}}$. In more detail, we combine a (possibly malleable) timed commitment with a non-malleable commitment to construct a commitment that is equivocal (*i.e.* can, using a trapdoor, be opened to a different value than the one committed to) and concurrently simulation-sound, *i.e.* retains its binding property even if the adversary sees equivocated commitments. We show that this suffices to replace the common reference string (CRS) of the UC-secure commitment scheme of Canetti and Fischlin [CF01] with coin-tosses, assuming that trapdoor permutations with dense public description [DP92] exist. The resulting composable commitment scheme is constant-round, black-box, in the plain model and makes use of standard polynomial-time and standard timed assumptions only. We note that our approach is conceptually different to recent results [FKPS21; KLX20; BDD$^+$21; BDD$^+$20] which define non-malleable or composable timed primitives and realize them using a trusted setup.

Due to the reusability of UC protocols, we can plug our construction into any UC protocol in the $\mathcal{F}_{\text{MCOM}}$-hybrid model while maintaining TLUC security. Using *e.g.* a variant of the MPC protocol of Hazay and Venkitasubramaniam [HV15], we are the first to obtain a composable constant-round, black-box and environmentally friendly general MPC protocol from standard polynomial-time and timed assumptions that does not impact the security of other protocols relying on (non-timed) polynomial-time hardness assumptions.

More generally, we can "bootstrap" the most efficient UC-secure protocol for a given task by realizing its setup within TLUC in the plain model. Depending on the setup, *e.g.* in the case of a CRS, the only overhead in this case consists of an input-independent preprocessing phase. This is not possible with previous approaches that feature composable MPC in the plain model.

3.1.1. Related Work

Informally, the very strong impossibility results for UC security [CF01; CKL03; PR08; KL11] imply that, unless an honest majority exists, UC security can only be achieved using some kind of trusted setup. Lindell [L03] has shown that the impossibilities are not due to the particular definition of UC security, but apply to general concurrent composability.

Ever since, there have been numerous attempts to circumvent these impossibility results at least partially by considering security notions that are weaker than standard UC security.

SPS Security, introduced by Pass [P03], considers simulators that may have a super-polynomial runtime, giving them an advantage over the polynomially-bounded environment at the expense of environmental friendliness and UC reusability.

While earlier approaches such as [P03; BS05] require (non-standard) super-polynomial hardness assumptions, newer approaches such as [LPV09; GGJS12; GKP18] require only standard polynomial-time hardness assumptions.

Due to the complexity asymmetry between environment and simulator, these constructions do not offer general composition. Thus, concurrent self-composability of SPS-secure protocols is often proven in a non-modular way. The transitivity of SPS security holds only with respect to protocols whose security is not "hurt" by the stronger simulator, *e.g.* protocols that are information-theoretically secure such as [IPS08]. Thus, (general) reusability of UC protocols is lost.

Lin, Pass, and Venkitasubramaniam [LPV09] have generalized the notion of UC security to $(\mathcal{C}_{\mathsf{env}}, \mathcal{C}_{\mathsf{sim}})$ security, where $\mathcal{C}_{\mathsf{env}}$ and $\mathcal{C}_{\mathsf{sim}}$ denote the complexity classes of environment resp. simulator. They present a construction for non-malleable zero-knowledge from UC puzzles that can be plugged into an appropriate general MPC protocol. For their construction in the plain model, [LPV09] assume simulators that run in non-uniform polynomial time while the environment runs in uniform polynomial time. However, the non-uniform simulation may impact the security of protocols that have started in the past. Also, if $\mathcal{C}_{\mathsf{sim}}$ is non-uniform polynomial-time, then the security notion is not meaningful for ideal functionalities that rely on uniform polynomial-time hardness assumptions.

Dachman-Soled *et al.* [DMRV13] have extended the work of [LPV09] by considering adaptive security. Starting with a UC puzzle, they construct a commitment scheme satisfying their new and strong notion of non-malleability from simulatable public-key encryption. This non-black-box and non-constant-round construction can then be plugged into an appropriate protocol, yielding secure composable general MPC with respect to adaptive corruptions.

Recently, Garg, Kiyoshima, and Pandey [GKP18] have presented a SPS-secure black-box oblivious transfer (OT) protocol from constant-round semi-honest OT and collision-resistant hash functions, *i.e.* standard polynomial-time hardness assumptions only. Their construction is secure against static corruptions and has a lower round complexity than other constant-round constructions such as [BDH$^+$17].

Angel-based Security and Environmental Friendliness. The weak composition properties of SPS security have subsequently been improved upon by notions where the simulator itself remains polynomially bounded, but is aided by some super-polynomial entity that is also available to the environment. Such frameworks include *Angel-based security* [PS04], or *UC with super-polynomial helpers* [CLP10]. [CLP10] construct a non-constant-round commitment scheme that is secure under chosen commitment attack (CCA) from one-way functions and use it to realize the ideal functionality for commitments. The extractor for the CCA commitment scheme features an important property called *k-robustness* (see also Definition 4.11). Informally, this means that an adversary $\mathcal{A}$ with access to a CCA oracle interacting with a k-round protocol can be replaced with an adversary $\mathcal{B}$ *without* access to the CCA oracle in an indistinguishable way. In other words, this means that the CCA oracle does not negatively affect the security of k-round protocols executed concurrently. By using this robustness property, their commitment scheme can be plugged into a k-round UC protocol ρ in the $\mathcal{F}_{\text{COM}}$-hybrid model.

This *round robustness* property has been generalized by Canetti, Lin, and Pass [CLP13a] to the property of *environmental friendliness*. The helper of [CLP13a] is environmentally friendly for protocols whose security is proven via black-box reductions to game-based cryptographic hardness assumptions with bounded polynomial round complexity.

Shielded Oracles. Broadnax *et al.* [BDH$^+$17] have introduced the notion of UC security with *shielded oracles* that lies strictly between SPS security and Angel-based security. Their construction for a composable commitment uses standard polynomial-time hardness assumptions only, is constant-round and black-box. While their notion is not environmentally friendly, they showed that the constructions can be plugged into a special class of UC-secure protocols without loss of security.

Other Models and Notions. There have been proposed a number of different models which enable (composable) MPC in the plain model. The *timing model* introduced by Kalai, Lindell, and Prabhakaran [KLP05] considers a communication network with time bounds and parties that have access to a local clock with little drift. There, non-constant-round non-black-box MPC secure under general composition is possible. This is done by *delaying* other protocols that are executed concurrently and incomparable to our approach.

The notion of *input indistinguishability*, first defined by Micali, Pass, and Rosen [MPR06] and generalized and strengthened by Garg *et al.* [GGJS12], is another security notion capturing concurrent self-composition that can be achieved in the plain model. However, the constructions of [MPR06; GGJS12] are non-black-box. Also, input indistinguishability is weaker than UC security.

Non-Malleable Time-Lock Puzzles and Commitments. Freitag *et al.* [FKPS21] have introduced the notion of *non-malleable time-lock puzzles* and timed commitments and present constructions in the random oracle model. Similar results have been obtained by Katz, Loss, and Xu [KLX20] in the algebraic group model. While both results can possibly be used as building blocks in our constructions, they are not in the plain model.

TARDIS and CRAFT. TARDIS [BDD$^+$21] extends the Generalized Universal Composability (GUC) framework [CDPW07] to include a notion of *abstract time* and *ticked functionalities* whose behavior can depend on the elapsed time. In this setting, universally composable abstractions of time-lock puzzles can be defined and realized in the random oracle model. We note that the goal of [BDD$^+$21] is different from ours. We use stand-alone-secure and possibly *malleable* timed primitives such as (malleable) timed commitments in order to achieve composability in the plain model. In contrast to TARDIS, we do not aim to define composable security notions for timed primitives. CRAFT [BDD$^+$20] realizes composable MPC in the TARDIS framework with additional guarantees such as output-independent abort, also relying on a random oracle.

3.1.2. Our Results

New Security Notion for Composable Security. The notion of UC security considers entities that are polynomially bounded and inherently unaware of other computations, apart from what is learned through communication. In particular, there is no notion of time. Thus, timed assumptions cannot be properly used in UC protocols. With TLUC security, we consider a variant of UC security that allows a party P to set up *timers* associated with a number of steps ℓ. At any point, P may query if the execution experiment in total (including the environment, adversary and other protocol parties) has performed ℓ or more steps. This allows the use of timed cryptographic primitives such as timed commitments in a meaningful way.

Similar to SPS security, our security notion is not closed under universal composition, but features the single-instance composition theorem (Theorem 3.3).

Environmental Friendliness. Very informally, *environmental friendliness*, introduced by Canetti, Lin, and Pass [CLP13a], deals with the problem of negative "side-effects" a protocol π may have on game-based properties of another protocol π' that runs *along-side* (where neither protocol is a subroutine of the other) and relies on polynomial-time hardness assumptions. Formally, this is captured in a stand-alone model for game-based security properties. Previous notions that feature general MPC in the plain model suffer from limited environmental friendliness because super-polynomial simulation, *e.g.* due to use of a super-polynomial helper, may break polynomial-time hardness assumptions of other protocols that run along-side, resulting in limited environmental friendliness. While not considered by the definition of environmental friendliness, giving the simulator non-uniform advice may hurt the security of (even non-uniformly) secure protocols or protocols that have been previously executed. Being a special case of UC security, TLUC security is fully environmentally friendly (Proposition 3.9). Also, TLUC security can be achieved with uniform simulation.

We note that the established notion does not consider *timed* game-based properties such as the timed hiding property of a timed commitment scheme. As such, our notion as well as *all* previous notions such as *e.g.* SPS security, Angel-based security and even UC security are not fully friendly in this respect.

UC Compatibility and Reusability. As *all* UC protocols retain their security under our notion (UC compatibility, Proposition 3.4) and TLUC simulators run in strict polynomial time, we can realize a UC-complete functionality $\mathcal{F}$ in TLUC and plug it into *any* existing UC-secure protocol making one subroutine call to $\mathcal{F}$ without loss of security (UC reusability, Corollary 3.5). This is not implied by environmental friendliness *per se*. In particular, we can use the best UC-secure protocol for a certain task and realize its setup within TLUC. This general reusability is not possible with other notions featuring composable MPC in the plain model as their UC reusability is limited.

As the simulation is always polynomial-time, (even uniformly only) computationally secure ideal functionalities are meaningful in our framework.

Composable Commitment Scheme in the Plain Model. Combining a timed commitment scheme and a non-timed commitment scheme that is secure under parallel chosen commitment attack (pCCA), we construct a non-malleable and partially simulatable coin-toss that is sufficient to "bootstrap" the CRS of a UC-secure commitment scheme such as the $\mathsf{UCC}_{\mathsf{OneTime}}$ scheme of Canetti and Fischlin [CF01] in the plain model. The resulting commitment scheme is concurrently composable and TLUC-realizes the ideal functionality for multiple commitments $\mathcal{F}_{\text{MCOM}}$ (Theorem 3.5). As the simulation is uniform and requires polynomial time only, π_{MCOM} does not hurt the security of any protocol using polynomial-time assumptions, including uniform ones.

Our protocol can also be adapted to other security notions, *e.g.* SPS security (see Section 3.6.1).

Composable Constant-Round General MPC in the Plain Model. Plugging our construction for $\mathcal{F}_{\text{MCOM}}$ into a variant of the general MPC protocol due to Hazay and Venkitasubramaniam [HV15], we obtain a constant-round black-box and environmentally friendly general MPC protocol from standard polynomial-time and standard timed assumptions in the plain model (Theorem 3.6). We remark that our results are in the static corruption setting and that we assume authenticated communication.

3.1.3. Outline

We first cover important definitions and technical aspects in Section 3.2. In Section 3.3, we introduce the notion of *timed simulation-soundness* for commitment schemes and present a construction. We continue with a presentation of TLUC security (Section 3.4), which is a variant of UC security that allows the use of timed assumptions and is fulfilled by our composable commitment scheme in the plain model (Section 3.5). Its security proof is presented in Section 3.6. Finally, we show how we can use this commitment scheme to achieve composable general MPC in Section 3.7.

3.2. Definitions

In the following, we introduce definitions used in subsequent sections.

3.2.1. Machine Model, Notion of Time

When considering polynomial-time hardness assumptions, the particularities of machine models rarely matter. This is because different (classical) machine models can be usually emulated by each other with polynomial runtime overhead or speedup. With polynomial time being closed under addition and multiplication, polynomial-time hardness assumptions do not become insecure if there is a machine model where some problem can be solved (polynomially) more efficient.

In this work, we consider timed primitives such as timed commitment schemes. For timed primitives, security often is only guaranteed against adversaries adhering to some kind of (concrete) runtime bound in a fixed machine model. In such a case, changing the machine model can make the difference between security and insecurity. This is obvious for stark differences, *e.g.* when going from a sequential to a parallel model of computation when considering timed assumptions that hold only against sequential adversaries. However, this problem also manifests with more subtle changes like allowing a larger alphabet for Turing machines, which may result in a linear speedup.

More problems arise during security reductions that require the emulation of Turing machines. Suppose that we want to show the security of some protocol π by using a ℓ-bounded timed assumption. We call ℓ the *timed security parameter*. In the security proof, the adversary $\mathcal{A}'$ against the timed assumption has to internally emulate the ℓ-bounded adversary $\mathcal{A}$ as well as (parts of) the protocol π. Just internally emulating the ℓ-bounded adversary may incur an overhead that does not allow the reduction to go through, because $\mathcal{A}'$ may always require more than ℓ steps due to its emulation overhead, even when just running the code of $\mathcal{A}$ and relaying messages. Additional overhead may occur *e.g.* for extracting the correct answer based on the internally emulated adversary's output. These caveats have to be accounted for.

Later on, we use timed primitives in the UC framework (*cf.* Section 3.4). While UC security can be stated using various machine models [C01], we adhere to the standard model of interactive Turing machines (ITMs). However, as *e.g.* the particular alphabet or the number of work tapes is left unspecified[1], so is the exact notion of runtime in that particular model. In order to argue about the security of timed assumptions in our security notion, we thus have to map the underspecified notion of runtime of interactive Turing machines as defined in the UC framework to the (possibly also underspecified) notion of runtime for the timed assumption. Following the Cobham-Edmonds thesis

[1]Newer versions of the UC framework such as [C20] explicitly allow multiple work tapes, allowing the emulation of other Turing machines with only additive overhead.

(see *e.g.* [G08]) or the extended Church-Turing thesis, we assume that this is always possible with a polynomial overhead or speedup in a classical setting, *i.e.* when not considering quantum computers.

For common machine models such as Turing machines, Boolean circuits or (parallel) random access machines, explicit emulation constructions and bounds for the overhead resp. speedup are known.

When constructing a protocol with security against $\ell(\kappa)$-bounded adversaries, we thus require the timed building blocks to be secure against adversaries with timed security parameter $\ell'(\ell(\kappa), \kappa)$[2] where ℓ' is a sufficiently large polynomial that accounts for possible runtime mismatches due to emulation overhead, reduction overhead or (polynomial) efficiency changes between machine models. As we do not want to make assumptions about the machine models being used, we do not explicitly specify ℓ'. However, as soon as all machine models and reductions are fixed, ℓ' is well-defined. Also, for our constructions, we show that ℓ' is sufficiently generic and *e.g.* is independent of the TLUC environment under consideration.

Note that the timed security parameter generally grows with increasing protocol nesting depth, similar to the tightness loss in standard reductions.

In our protocols, we use `timer` messages parameterized by an ID *id* to allow protocol parties later check if more steps than allowed by the timed security parameter ℓ have been elapsed by sending a message $(\texttt{notify}, \mathit{id})$. If the answer is $(\texttt{notify}, \mathit{id}, 1)$, then more than ℓ steps have passed and we say that the "timer has timed out" or "expired". Conversely, $(\texttt{notify}, \mathit{id}, 0)$ denotes that the timer has not expired. Later on, we will only consider adversaries (or environments) that handle such messages correctly.

As the default machine model and execution experiment of UC are inherently sequential, we refer to *computation steps* instead of *runtime*, as the latter may capture many steps performed in parallel, which we want to count individually.

3.2.2. Commitment Schemes

In the following, we consider additional properties of commitment schemes that are exclusive to this chapter.

Often, we are interested in so-called *tag-based* commitment schemes [DDN00; PR05a], which are defined as follows:

Definition 3.1 (Tag-Based Commitment Scheme)**.** A commitment scheme COM is *tag-based* if the committer C and the receiver R additionally receive a tag *tag* from a tag space T as common input.

While Definition 3.1 is only a syntactic extension of Definition 2.8, the inclusion of tags allows the definition of additional security properties that are of interest in a setting where multiple commitments are performed concurrently and security guarantees going beyond the standard binding and hiding properties are required. In particular, it may be

[2]In order to capture the setting where $\ell(\kappa)$ is constant but *e.g.* the reduction overhead depends on κ, we parameterize ℓ' with both values.

desirable for a commitment scheme to be *non-malleable*. Informally, this means that a commitment performed with a tag tag_1 cannot be transformed into a commitment with a different tag tag_2. If the tags are guaranteed to be unique throughout the execution, the non-malleability property prevents, for example, an adversary to commit to a value related to the value inside a commitment it concurrently performs with an honest party. This independence is neither guaranteed by the binding nor the hiding property.

As the first tag-based property, we consider a stronger notion of the hiding and non-malleability property called security under *parallel chosen commitment attack (pCCA)* [K14; BDH+17; BFMR18]. In the pCCA hiding experiment, the adversary may additionally interact with an (inefficient) oracle $\mathcal{O}_{\mathsf{pCCA}}$ to perform an unbounded number of commitments *in parallel*, with $\mathcal{O}_{\mathsf{pCCA}}$ acting as receiver. After all commit phases with $\mathcal{O}_{\mathsf{pCCA}}$ have finished, $\mathcal{O}_{\mathsf{pCCA}}$ outputs, for each commitment, the unique value committed to. If no such value exists, a special symbol $\perp$ is returned for this commitment. The challenge commitment where the adversary acts as receiver must remain hiding, even with access to $\mathcal{O}_{\mathsf{pCCA}}$.

Definition 3.2 (pCCA Security)**.** For a commitment scheme $\mathsf{COM} = \langle \mathsf{C}, \mathsf{R} \rangle$, the random variable $\mathrm{Exp}^{\text{pCCA-Hiding}}_{\mathcal{A},\mathsf{COM},\mathcal{O}_{\mathsf{pCCA}}}(\kappa, z)$ for the pCCA hiding experiment is defined as follows:

1. Run $\mathcal{A}^{\mathcal{O}}_{\mathsf{pCCA}}$ on input $(1^\kappa, \texttt{find}, z)$ and obtain $(m_0, m_1, state)$.
2. Sample a uniformly random bit $b \xleftarrow{\$} \{0, 1\}$.
3. If $|m_0| \neq |m_1|$, return b.
4. Otherwise, obtain $b' \leftarrow \mathrm{out}_{\mathcal{A}}\langle \mathsf{C}(m_b), \mathcal{A}^{\mathcal{O}_{\mathsf{pCCA}}}(\texttt{guess}, state)\rangle(1^\kappa, \texttt{commit})$.
5. If $b = b'$, output 1. Otherwise, output 0.

The (stateful) oracle $\mathcal{O}_{\mathsf{pCCA}}$ acts as honest receiver R for multiple sessions of COM *in parallel*. When all commit phases have finished, the oracle returns the unique values committed to. If no such unique value exists, a special symbol $\perp$ is output for these commitments. An adversary $\mathcal{A}$ is valid if it eventually outputs a bit and never interacts with $\mathcal{O}_{\mathsf{pCCA}}$ on the challenge tag. We say that COM is *pCCA-secure* or *pCCA-hiding* if for every valid PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for every $\kappa \in \mathbb{N}$ and every $z \in \{0, 1\}^*$, it holds that $\mathrm{Adv}^{\text{pCCA-Hiding}}_{\mathcal{A},\mathsf{COM},\mathcal{O}_{\mathsf{pCCA}}}(\kappa, z) = |\Pr[\mathrm{Exp}^{\text{pCCA-Hiding}}_{\mathcal{A},\mathsf{COM},\mathcal{O}_{\mathsf{pCCA}}}(\kappa, z) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\kappa)$. The probability is over the coins of $\mathcal{A}$ and the coins of the experiment, including the oracle.

We continue with a definition of trapdoor commitment schemes, *i.e.* commitment schemes that a) are not binding if a secret trapdoor is known and b) fulfill the trapdoor property, meaning that a (possibly malicious) receiver cannot distinguish between an honestly created (and opened) commitment and a commitment created (and opened) using the trapdoor. Looking ahead, we will need these properties to hold even if the adversary is given access to certain oracles, *e.g.* a (p)CCA oracle. However, previous definitions of trapdoor commitments such as the ones of [MY04; GMY03] do not provide the adversary with such oracles. In the following, we define a stronger variant of the trapdoor property where the adversary is additionally provided with an oracle.

Definition 3.3 (Trapdoor Commitment Scheme)**.** A commitment scheme $\mathsf{TRAPCOM} = (\mathsf{C}, \mathsf{R}, \mathsf{C}_{\text{trap}})$ is called *trapdoor* for oracle $\mathcal{O}$ and message space M if $\langle \mathsf{C}, \mathsf{R}\rangle$ and $\langle \mathsf{C}_{\text{trap}}, \mathsf{R}\rangle$ are commitment schemes with message space M such that for every valid PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for every $\kappa \in \mathbb{N}$ and every $z \in \{0,1\}^*$, it holds that $\mathrm{Adv}^{\mathrm{TD}}_{\mathcal{A},\mathsf{TRAPCOM},\mathcal{O}}(\kappa, z) = |\Pr[\mathrm{Exp}^{\mathrm{TD}}_{\mathcal{A},\mathsf{TRAPCOM},\mathcal{O}}(\kappa, z) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\kappa)$. The probability is over the coins of $\mathcal{A}$ and the coins of the TD experiment. The random variable $\mathrm{Exp}^{\mathrm{TD}}_{\mathcal{A},\mathsf{TRAPCOM},\mathcal{O}}$ is defined as follows:

1. Sample $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
2. Run $\mathcal{A}$ on input $(1^\kappa, z)$. $\mathcal{A}$ interacts with the experiment by first sending $(\texttt{start}, \mathit{tag}, v)$ to start the commit phase of $\mathsf{TRAPCOM}$, acting as receiver. If $b = 0$, the experiment runs the code of the honest committer C on input $(1^\kappa, \mathit{tag}, v)$. If $b = 1$, the experiment runs the code of the trapdoor committer C_{trap} on input $(1^\kappa, \mathit{tag}, |v|)$. After the commit phase has finished, the unveil phase is performed. At any time, $\mathcal{A}$ may interact with the (possibly stateful) oracle $\mathcal{O}(\cdot,\cdot)$ which takes a tag as first argument. Finally, $\mathcal{A}$ outputs a bit b'.
3. Output 1 if $b = b'$ and 0 otherwise.

An adversary $\mathcal{A}$ is called *valid* if it never queries $\mathcal{O}$ with a tag that has the tag of the challenge commitment as prefix, if $v \in M$ and $b' \in \{0,1\}$.

If $\mathrm{Adv}^{\mathrm{TD}}_{\mathcal{A},\mathcal{O},\mathsf{TRAPCOM}}$ is negligible for unbounded adversaries, then $\mathsf{TRAPCOM}$ is called *statistically trapdoor*. If $\mathcal{A}$ is unbounded and its advantage is 0, then $\mathsf{TRAPCOM}$ is called *perfectly trapdoor*.

We may parameterize trapdoor commitment schemes with additional parameters related to timed security properties. However, as the trapdoor property is a polynomial-time property, we have chosen to omit these parameters here.

It directly follows that a commitment scheme satisfying the above notion is also hiding.

3.2.3. Timed Commitment Schemes

Boneh and Naor [BN00] have introduced the notion of *timed commitment schemes*. Instead of the hiding property holding against all polynomial-time adversaries, a (T, ℓ, ε)-timed commitment scheme guarantees the hiding property to hold only for some bound of steps ℓ performed by an adversarial receiver, except with probability ε.

However, the (ℓ, ε)-hiding property does not guarantee that there exists a value $T \in \mathbb{N}$ such that a valid timed commitment can be opened "forcefully" in at most $T > \ell$ steps. To this end, the definition of [BN00] also requires the existence of a `forced-open` algorithm that runs in time T, takes the transcript of a successful commit phase and outputs the unique value $v \in M$ committed to, where M is the message space of the commitment scheme. In other words, in addition to the binding property, a malicious committer must not be able to open its commitment to a value that is inconsistent with the output of the `forced-open` algorithm. This extractability is crucial for our simulation later on, as it guarantees that simulators can extract timed commitments in polynomial time (if T is bounded by a polynomial in κ).

In the definition of [BN00], timed commitment schemes have to exhibit a *soundness property* which requires that at the end of the commit phase, the receiver is "convinced" that running the `forced-open` algorithm will produce the value v committed to. While not formally defined, the definition of [BN00] also requires valid commitments to be efficiently recognizable by the receiver.

Looking ahead to our construction, we do not need valid timed commitments to be efficiently recognizable. In particular, we can deal with the over-extraction of invalid commitments, *i.e.* the case where `forced-open` outputs a value $v \in M$, even if the commitment cannot be unveiled. We call this property *weak extractability* and will account for this in the following definition.

Also, the hiding property informally described in [BN00] seems to be relatively weak, considering honestly created commitments only. Moreover, the adversary's steps are only counted after it is provided the transcript of a successful commit phase. Our definition of timed hiding (Definition 2.9) is standard and stronger in the sense that the commitment *receiver* may act maliciously. Also, we count the adversary's steps from the very beginning on. It is easy to see that the timed commitment scheme in [BN00] satisfies this stronger notion.

With [BN00] not giving a formal definition, we define weakly extractable timed commitment schemes as follows, omitting the parameter ε for the adversary's advantage in the hiding game. Instead, we only consider timed commitment schemes where an adversary's advantage is negligible.

Definition 3.4 (Weakly Extractable Timed Commitment Scheme)**.** A tuple of ITMs $\mathsf{TCOM} = \langle \mathsf{C}, \mathsf{R} \rangle$ is called a $(T(\kappa), \ell(\kappa))$-*weakly extractable timed commitment scheme* with message space M if $\langle \mathsf{C}, \mathsf{R} \rangle$ is a ℓ-hiding commitment scheme for which there exists a deterministic algorithm `forced-open` that, given a transcript c of a successful commit phase, outputs the unique value $v \in M$ committed to in at most T steps.

In the following, we often explicitly consider ℓ to be part of the common input.

We say that TCOM is *perfectly correct* if for every $\kappa \in \mathbb{N}$ and every $v \in M$,

$$\begin{aligned}\Pr[v^\star = v' = v \mid (\mathit{view}_\mathsf{C}, \mathit{view}_\mathsf{R}, c) \leftarrow \mathrm{out}\langle \mathsf{C}(v), \mathsf{R}(\varepsilon)\rangle(1^\kappa, \mathtt{commit}, \ell),\\ v' \leftarrow \mathrm{out}_\mathsf{R}\langle \mathsf{C}(\mathit{view}_\mathsf{C}), \mathsf{R}(\mathit{view}_\mathsf{R})\rangle(\mathtt{unveil}), v^\star = \mathtt{forced\text{-}open}(c)] = 1\end{aligned}$$

The perfect correctness can be naturally relaxed to statistical correctness. We also assume that M is efficiently recognizable and that the receiver rejects an opening to a value $v \notin M$.

We say that TCOM is *perfectly binding* and *weakly extractable* if for every (malicious) committer C^*, every $\kappa \in \mathbb{N}$ and every $z \in \{0,1\}^*$, it holds that

$$\begin{aligned}\Pr[&v^\star = v' \mid (\mathit{view}_{\mathsf{C}^*}, \mathit{view}_{\mathsf{R}}, c) \leftarrow \mathrm{out}\langle \mathsf{C}^*(z), \mathsf{R}(\varepsilon)\rangle(1^\kappa, \mathtt{commit}, \ell),\\ &v' \leftarrow \mathrm{out}_{\mathsf{R}}\langle \mathsf{C}^*(\mathit{view}_{\mathsf{C}^*}), \mathsf{R}(\mathit{view}_{\mathsf{R}})\rangle(\mathtt{unveil}), v^\star = \mathtt{forced\text{-}open}(c) \land v' \in M] = 1\end{aligned}$$

While the aforementioned properties do not state any requirements for the output of `forced-open` on invalid commitments (*i.e.* allow over-extraction), it implies the soundness requirement of [BN00] for valid commitments.

Definition 3.4 is not concerned with the committer's runtime, which may depend on all parameters, in particular T and ℓ. This is important for (proving) security properties that consider more than one commitment, *e.g.* the timed simulation-soundness (Definition 3.8).

Boneh and Naor [BN00] also present a constant-round construction based on the generalized Blum-Blum-Shub (BBS) assumption [BN00] that does not use non-black-box techniques. Also, their construction admits a super-polynomial gap between the number of steps needed to perform the commitment and the number of steps ℓ the commitment is secure against.

While [BN00] consider a machine model that admits parallel computations, we consider (weaker) sequential models of computation only.

Recently, Freitag *et al.* [FKPS21] and Katz, Loss, and Xu [KLX20] have re-visited timed commitment schemes, providing formal definitions and new constructions. However, as they consider (non-interactive) timed commitment schemes with setups, their definitions are not easily applicable to our setting.

Timed commitments can also be constructed by combining *sequential functions* [MMV13] and universal hash functions. However, such a construction has the drawback that both commit and unveil phase are computation-intensive. Still, it suffices for a feasibility result with a symmetric assumption.

Looking ahead to our constructions, we remark that using timed commitments with non-malleability properties in the plain model will not lead to easier definitions or proofs due to the power of the simulator. We leave it as an open question whether there are advantages if the simulator is restricted like *e.g.* in the Angel-based setting.

3.2.4. Trapdoor Pseudorandom Generators

In our construction in Section 3.5, we will use so-called *trapdoor pseudorandom generators (PRGs)*. Being very similar to ordinary PRGs, they additionally allow to efficiently determine whether a value is in the range of the PRG with the help of a trapdoor.

We propose the following definition.

Definition 3.5 (Trapdoor PRG)**.** A *trapdoor PRG* PRG is comprised of four PPT algorithms $(\mathsf{Gen}, \mathsf{TGen}, \mathsf{PRG}, \mathsf{TCheck})$ such that the following conditions hold:

1. On input 1^κ, Gen outputs a public key $pk \in \{0,1\}^{l(\kappa)}$, where l is a polynomial in κ denoting the key length.

2. On input 1^κ, TGen outputs a key pair $(pk, sk) \in \{0,1\}^{l(\kappa)} \times \{0,1\}^{l(\kappa)}$, where l is a polynomial in κ denoting the key length.
3. On input (pk, r) with pk in the range of $\mathsf{Gen}(1^\kappa)$ and $r \in \{0,1\}^\kappa$, PRG outputs a value $r' \in \{0,1\}^{e(\kappa)}$, where e is a polynomial in κ denoting the PRG's output length, subject to the condition that $e(\kappa) > \kappa$ for every $\kappa \in \mathbb{N}$.
4. On input (pk, sk, r') with (pk, sk) in the range of $\mathsf{TGen}(1^\kappa)$, TCheck outputs 1 if r' is in the range of $\mathsf{PRG}(pk, \cdot)$ and 0 otherwise.
5. The ensembles $\{\mathsf{PRG}(pk, r)\}_{\kappa\in\mathbb{N}, pk\leftarrow\mathsf{Gen}(1^\kappa), r\xleftarrow{\$}\{0,1\}^\kappa}$ and $\{r^*\}_{\kappa\in\mathbb{N}, r^*\xleftarrow{\$}\{0,1\}^{e(\kappa)}}$ are computationally indistinguishable (pseudorandomness).
6. The ensembles $\{\mathsf{Gen}(1^\kappa)\}_{\kappa\in\mathbb{N}}$ and $\{pk\}_{\kappa\in\mathbb{N},(pk,sk)\leftarrow\mathsf{TGen}(1^\kappa)}$ are computationally indistinguishable.

Similar to trapdoor permutations with dense public descriptions (Definition 2.7), we define the analogous property for PRGs.

Definition 3.6 (PRG with Dense Public Description)**.** A PRG has a *dense public description* if the following condition holds:

1. The ensembles $\{\mathsf{Gen}(1^\kappa)\}_{\kappa\in\mathbb{N}}$ and $\{r\}_{\kappa\in\mathbb{N}, r\xleftarrow{\$}\{0,1\}^{l(\kappa)}}$ are computationally indistinguishable.

We recall the following proposition, implicit in [CF01].

Proposition 3.1. *If trapdoor permutations exist, then trapdoor PRGs exist as well.*

Specifically, Canetti and Fischlin [CF01] propose the Blum-Micali-Yao construction [Y82; BM82] with a trapdoor permutation instead of a one-way permutation. It is easy to see that if the trapdoor permutation has a dense public description, the same holds for the resulting PRG.

3.3. Timed Simulation-Sound Commitment Schemes

Looking ahead to our construction of a composable commitment scheme (Section 3.5), we need a commitment scheme that is equivocal for a polynomial-time simulator. At the same time, commitments created by a malicious committer that adheres to certain time-related rules and bounds must remain binding sufficiently long, even if the malicious committer sees equivocated commitments. To this end, we first define the security notion of *timed simulation-soundness*. Also, we present the construction SSCOM (where SS denotes *simulation-sound*) that combines a possibly malleable timed commitment scheme with a non-timed commitment scheme that is secure under parallel chosen commitment attack [K14; BDH+17; BFMR18] and satisfies the notion of timed simulation-soundness.

3.3.1. Timed Simulation-Soundness

Based on the established notion of simulation-soundness [MY04; GMY03] and inspired by the non-malleability notion of Dachman-Soled *et al.* [DMRV13], we define a concurrent and timed variant of simulation-soundness that is suitable for commitments where the binding property only holds temporarily (Definition 3.8). Intuitively, this *timed* simulation-soundness ensures that commitments produced by a malicious committer remain binding for a *temporarily* bounded adversary even if it concurrently receives equivocated commitments.

While somewhat similar to the notion of *non-malleability with respect to unveil* or *opening* or *decommitment* ([DIO98; PR05b; OPV08; FF11]), our definition is stronger in the sense that commit and unveil phases may be interleaved in the experiment (similar to UC commitments or the definition of [DMRV13]).

The Experiment. In the experiment for *timed simulation-soundness*, a man-in-the-middle adversary acts as receiver in an unbounded number of instances ("left sessions") of some trapdoor commitment scheme (*cf.* Definition 3.3). The adversary starts left sessions by providing a tag of its choice, along with an efficiently samplable and length-normal (*cf.* Definition 3.7) distribution. Only considering distributions facilitates easier proofs and more general definitions and is sufficient for our application. In each left session, the code of the trapdoor committer $\mathsf{C}_{\mathrm{trap}}$ is executed. After the commit phase of a session has finished, the adversary may, at some point of its choice, start the unveil phase. At its onset, a value from the provided distribution is sampled and unveiled by the trapdoor committer.

In addition, the adversary acts as committer in one session ("right session"), again using a tag of its choice that must be unique compared to all other tags that will eventually be used in the experiment. The scheduling between all sessions and their messages is fully controlled the adversary.

When the commit phase of the single right session has finished, the experiment determines the value committed to. The commitment scheme is timed simulation-sound if the adversary cannot unveil its single commitment to a value different from the committed one, even when presented with equivocated commitments.

Timer-Related Parameters. In our setting, we do not consider simulation-soundness against arbitrary polynomial-time adversaries. Indeed, our construction SSCOM is (intentionally) not simulation-sound or even binding for polynomial-time adversaries: If a corrupted receiver manages to break a timed commitment it receives from the (honest) sender early enough, the constructed commitment becomes equivocal. In our setting, protocol parties may set up *timers* and inquire at some point whether the timer has expired. The timed simulation-soundness experiment is thus parameterized with a timed security parameter ℓ. This timed security parameter denotes how many steps experiment and adversary may perform before a timer set up by the honest receiver in the right session is considered to have timed out. If no timeout occurs, the binding property of the single right commitment should hold, even if left commitments are equivocated.

Timed simulation-sound commitments that use timed building blocks such as timed commitment schemes must choose their timed security parameter ℓ' relative to ℓ. To account for reduction overhead, *e.g.* to the timed hiding property of a timed commitment scheme, ℓ' must be chosen sufficiently large. As the reduction overhead may depend on the security parameter κ but $\ell(\kappa)$ might be constant, ℓ' is also parameterized with κ.

Depending on the construction, increasing ℓ may lead to the timer *always* expiring, *e.g.* because a sub-protocol protected by the timer requires more than ℓ steps to execute (*e.g.* the commit phase of a timed commitment scheme, which may take longer for larger ℓ) for some values of ℓ. In this case, proving security becomes trivial as the adversary cannot win the game. However, this also implies that scheme is secure in this case. When using appropriate building blocks, *e.g.* non-interactive timed commitments or a timed commitment scheme with a sufficiently large gap (for example, the scheme of [BN00] has an exponential gap between the time needed to create the commitment and its timed security), this problem does not occur for sufficiently large ℓ'.

In order to notify parties about timeouts, we require the adversary to obey the following rules: When receiving a message $(\texttt{notify}, id)$ for some ID id, it must immediately answer $(\texttt{notify}, id, 1)$ if it has previously received $(\texttt{timeout}, id)$ and the whole experiment, including the adversary and *honest* committers in left sessions, has performed ℓ or more steps, where ℓ is the timed security parameter. For our construction, this can be easily computed as the runtime of the involved algorithms do not depend on their internal randomness or secrets. If an exact calculation is not possible, the adversary must use an appropriate upper bound.

This is in contrast to *e.g.* Definition 2.9 where only the steps of the adversary are counted. There, this is possible as only one commitment session is considered. Here, we consider an unbounded number of sessions. In a reduction to some timed property, all the left sessions will have to be emulated by the reduction adversary, counting against its time limit in the reduction.

As the guarantees of timed cryptographic assumptions are only for honest parties, the experiment does not answer `notify` messages (from the adversary).

In real life, one can of course not expect that a possibly malicious party obeys these rules. However, if a timed primitive is believed to be secure for *e.g.* several days considering the computation power available to the other party, assuming a timeout after, say, one minute, should be sufficiently secure.

Relationship to Other Non-Malleability Notions. Similar to the simulation-based non-malleability notion of [DMRV13], security must hold if the commit and unveil phases on the left side are interleaved with the right session. However, in contrast to [DMRV13], we do not require the commitment on the right side to be concurrently extractable and also do not consider adaptive corruptions, leading to a different security notion. Moreover, the notion in [DMRV13] does not consider any timed security properties whatsoever.

Formal Definition. First, we define length-normal probability distributions as distributions where all elements of the sample space are of equal length.[3]

Definition 3.7 (Length-Normal Probability Distribution)**.** Let $\mathcal{D}$ be a probability distribution over $\{0,1\}^*$ with sample space Ω. $\mathcal{D}$ is called *length-normal* if for every $x, y \in \Omega$, it holds that $|x| = |y|$. Let $|\mathcal{D}|$ denote $|x|$ for $x \in \Omega$.

An example for a length-normal distribution is the uniform distribution $\mathcal{U}_n$ over $\{0,1\}^n$ with $|\mathcal{U}_n| = n$.

Definition 3.8 (Timed Simulation-Soundness)**.** A trapdoor commitment scheme $\mathsf{TRAPCOM}$ with message space $M \subseteq \{0,1\}^*$ is called $\ell(\kappa)$*-timed simulation-sound* if for every legal PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for every $\kappa \in \mathbb{N}$ and for every $z \in \{0,1\}^*$, it holds that

$$\mathrm{Adv}^{\mathrm{SIMSOUND}}_{\mathcal{A},\mathsf{TRAPCOM}}(\kappa, \ell(\kappa), z) = \Pr[\mathrm{Exp}^{\mathrm{SIMSOUND}}_{\mathcal{A},\mathsf{TRAPCOM}}(\kappa, \ell(\kappa), z) = 1] \leq \mathsf{negl}(\kappa)$$

where the probability is over the random coins of the experiment and the adversary. An adversary $\mathcal{A}$ is called *legal* if i) it immediately sends the message $(\texttt{notify}, id, 1)$ after receiving $(\texttt{notify}, id)$ and the experiment (including the adversary) has performed more than or equal to $\ell(\kappa)$ steps after having received a message $(\texttt{timer}, id)$[4], where steps performed by the committer on left sides are counted as by the honest committer C and ii) $\mathcal{A}$ sends `commit-left` messages only parameterized with efficiently samplable and length-normal distributions (*cf.* Definition 3.7) where the sample space Ω is a subset of the message space M and iii) the tag used in the right commitment is never used in a left commitment.

The random variable $\mathrm{Exp}^{\mathrm{SIMSOUND}}_{\mathcal{A},\mathsf{TRAPCOM}}(\kappa, \ell(\kappa), z)$ is defined as follows:

1. Run the adversary $\mathcal{A}$ on input $(1^\kappa, \ell(\kappa), z)$.
2. Upon receiving $(\texttt{commit-left}, tag, \mathcal{D}_{tag})$ from the adversary: Start the commit phase of $\mathsf{TRAPCOM}$ with common input $(1^\kappa, \texttt{commit}, tag, \ell(\kappa))$, acting as trapdoor committer $\mathsf{C}_{\mathrm{trap}}$ with private input $|\mathcal{D}_{tag}|$, unless there already is a session with tag *tag*.
3. Upon receiving $(\texttt{commit-right}, tag)$ from the adversary: Start the commit phase of the right session with common input $(1^\kappa, \texttt{commit}, tag, \ell(\kappa))$, acting as honest receiver R, unless the right session already exists or there is a left session with tag *tag*. Let $v' \in M$ denote the value committed to in the right session. If no such unique value exists, set $v' = \perp$.
4. Upon receiving $(\texttt{unveil-left}, tag)$ from the adversary: Sample $v_{tag} \leftarrow \mathcal{D}_{tag}$ and start the unveil phase of the i-th left session with common input $(\texttt{unveil}, tag)$ and private input v_{tag} for the trapdoor committer, unless the commit phase with tag *tag* has not finished or the unveil phase has already started.

[3]When considering an appropriate encoding, the definition can be extended to *e.g.* group elements.
[4]We assume unique timer IDs within a protocol throughout this work.

5. Upon receiving (`unveil-right`) from the adversary: Start the unveil phase of the right session with common input (`unveil`, *tag*), acting as honest receiver where *tag* is the tag specified in the commit phase. Let u denote the value accepted by the receiver or $\perp$ in case of an abort.

6. Upon receiving (`message`, tag, m) from the adversary, forward the message m to the session with tag *tag*. Conversely, forward messages to the adversary.

7. After the right unveil phase has finished, output 1 if the receiver in the right session has accepted and $u \neq v' \wedge u \neq \perp$, *i.e.* if the adversary has unveiled the commitment in the right session to a value different from the extracted one. Otherwise, output 0.

For the sake of brevity, we also say that a commitment scheme fulfilling the above definition is $\ell(\kappa)$-simulation-sound or timed simulation-sound, omitting the timed security parameter.

Like [DMRV13], we call an adversary that wins the above experiment with at most negligible probability *non-abusing*, *i.e.* if its commitments remain binding even when presented with equivocated commitments.

Remark 3.1. Note that Definition 3.8 does not specify how the value committed to is determined. If the commitment scheme is statistically binding, the value committed to is defined with overwhelming probability. In other cases, such as our commitment scheme SSCOM, this is not possible, because it is equivocal by definition. For SSCOM, we will thus give an explicit description of an inefficient algorithm to determine the value committed to.

For commitment schemes that are statistically hiding, the case is more complicated as there is no (statistical) value committed to by definition. However, the *computational* value committed to can often be determined by rewinding the committer. We leave a general definition supporting this case as future work and refer the reader to Section 4.3 for discussions and definitional challenges.

Remark 3.2. While the binding property and simulation-soundness are related, a (timed) binding commitment scheme is not necessarily (timed) simulation-sound. In particular, this may be the case if the equivocation is performed by an entity for which the binding property does not hold, *e.g.* due to its computational resources.

3.3.2. The Construction SSCOM

In the following, we present the construction SSCOM (Construction 1) for a timed simulation-sound string commitment scheme, which is based on the commitment scheme due to Broadnax *et al.* [BDH+17], which is inspired by the construction of Damgård and Scafuro [DS13]. Roughly, the scheme presented in [BDH+17] works as follows: Committer and receiver perform a commitment to a random index vector $I \in \{0,1\}^\kappa$ chosen by the receiver. They then perform 2κ commitments to pair-wise shares of the secret. In the unveil phase, the committer first sends the shares without unveiling the share commitments. Then, the receiver unveils the commitment to I. Finally, the committer unveils the share commitments indexed by I, while the other commitments remain unopened. If the commitment scheme used for I is extractable, the constructed commitment is equivocal. As inconsistent share commitments remain unopened and hiding, a malicious receiver cannot distinguish between an equivocated and an honest commitment. In order to achieve concurrent security, we require the share commitment scheme to be pCCA-secure (Definition 3.2).

In contrast to the original construction of [BDH$^+$17], we use a timed commitment scheme for the commitment to the index vector I, which allows polynomial-time equivocation of SSCOM commitments. Also, we move this timed commitment to I to the end of the commit phase. Looking ahead, the receiver will only accept an opening if the timer has not expired until the committer has sent the shares in the unveil phase.

For the sake of simpler proofs, we assume that the commitment scheme for the shares is perfectly binding. However, this requirement can be relaxed to statistically binding.

To facilitate easy integration with our composable commitment scheme (Section 3.5) and the timed simulation-soundness definition (Definition 3.8), SSCOM includes explicit messages to set up timers and to check if they have expired. Again, the party answering the timer status inquiry checks if both parties have performed ℓ or more steps since the timer has been set up and answers accordingly. In the simulation-soundness experiment, the answer is given by the adversary that is required to answer truthfully. It would have been possible to only count steps by the party that has *not* set up the timer. However, counting the steps of both parties is more consistent with our other definitions and more convenient in reductions.

Construction 1 (Timed Simulation-Sound Commitment Scheme SSCOM)**.** Parameterized with a security parameter κ, a timed security parameter $\ell(\kappa)$, a commitment scheme $\mathsf{COM}_{\mathsf{pCCA}}$ with message space $M \supseteq \{0,1\}^\kappa$ and a $(T, \ell'(\ell(\kappa), \kappa))$-weakly extractable timed commitment scheme TCOM with message space $M' \supseteq \{0,1\}^\kappa$.

Commit Phase. On common input $(1^\kappa, \texttt{commit}, \mathit{tag}, \ell(\kappa))$, the committer C and receiver R interact as follows:

1. The committer C creates 2κ shares $s_{1,0}, s_{1,1}, \ldots, s_{\kappa,0}, s_{\kappa,1}$ of its private input v by sampling $s_{m,0} \xleftarrow{\$} \{0,1\}^{|v|}$ and setting $s_{m,1} = v \oplus s_{m,0}$, $m = 1, \ldots, \kappa$.

2. For $m = 1, \ldots, \kappa$, $n = 0, 1$, C and R start 2κ instances of $\mathsf{COM}_{\mathsf{pCCA}}$ on common input $(1^\kappa, \texttt{commit}, (\mathit{tag}, m, n))$ in parallel. The committer's private input in the instance with tag (tag, m, n) is $s_{m,n}$.

3. The receiver R samples an index vector $I \xleftarrow{\$} \{0,1\}^\kappa$ and sends $(\texttt{timer}, \mathit{tag})$ to C. Then, C and R start an instance of TCOM with common input $(1^\kappa, \texttt{commit}, \ell'(\ell(\kappa), \kappa))$. The receiver R of SSCOM acts as committer with private input I in the execution of TCOM.

Unveil Phase. On common input $(\texttt{unveil}, \mathit{tag})$, the committer C and receiver R interact as follows:

1. The committer C sends the shares $(s_{1,0}, \dots, s_{\kappa,1})$ to the receiver R.
2. The receiver R sends $(\texttt{notify}, \mathit{tag})$ to C, which R answers with $(\texttt{notify}, \mathit{tag}, b)$ where $b = 1$ if committer and receiver have spent more than or equal to $\ell(\kappa)$ steps since the timer has been set up. Otherwise, $b = 0$ indicates that less than $\ell(\kappa)$ steps in total have elapsed. If C answers with $(\texttt{notify}, \mathit{tag}, 1)$, R aborts. Otherwise, the receiver checks that $s_{1,0} \oplus s_{1,1} = \dots = s_{\kappa,0} \oplus s_{\kappa,1}$ and aborts if this does not hold. Then, C and R perform the unveil phase of TCOM. The committer C also ensures that the TCOM commitment is extractable (to the value I) in at most T steps, *e.g.* by using the `forced-open` algorithm. If this check fails, the committer C aborts.
3. Committer and receiver perform κ unveil phases of $\mathsf{COM}_{\mathsf{pCCA}}$ in parallel as follows: For $m = 1, \dots, \kappa$, the commitment to $s_{m,I[m]}$ with tag $(\mathit{tag}, m, I[m])$ is unveiled. Let $s'_{m,I[m]}$ denote the unveiled value of the commitment with tag $(\mathit{tag}, m, I[m])$.
4. After all unveil phases have finished, the receiver checks that $s'_{m,I[m]} = s_{m,I[m]}$, $m = 1, \dots, \kappa$. If this holds, the receiver outputs $s_{1,0} \oplus s_{1,1}$. Otherwise, it aborts.

Algorithm of the Trapdoor Committer $\mathsf{C}_{\mathrm{trap}}$.

1. On private input l in the commit phase, commit honestly to 0^l.
2. On private input $v \in \{0,1\}^l$ in the unveil phase, extract the (timed) TCOM commitment using the `forced-open` algorithm to obtain the index vector I. If `forced-open` fails, sample $I \xleftarrow{\$} \{0,1\}^\kappa$ uniformly at random. For $m = 1, \dots, \kappa$, send $s_{m,1-I[m]} = v \oplus s_{m,I[m]}$ as shares that will not be unveiled. Continue the unveil phase like the honest committer. If the timed commitment to I is opened to a value different from the extracted one, abort.

Proof of the Trapdoor Property of SSCOM.

We first prove that SSCOM is trapdoor for the pCCA oracle of $\mathsf{COM}_{\mathsf{pCCA}}$.

Theorem 3.1. *Let $\mathsf{COM}_{\mathsf{pCCA}}$ be a pCCA-secure and perfectly binding commitment scheme and let TCOM be a perfectly binding and weakly extractable timed commitment scheme, both with message space $\{0,1\}^\kappa$. Then, the commitment scheme SSCOM is a trapdoor commitment scheme with message space $\{0,1\}^\kappa$ for the pCCA oracle of $\mathsf{COM}_{\mathsf{pCCA}}$.*

Proof. As we assume TCOM to be perfectly binding and weakly extractable, $\mathsf{C}_{\mathrm{trap}}$ will never abort due to an inconsistent I and we omit these straight-forward reductions.

We prove Theorem 3.1 by a reduction to the pCCA hiding property of $\mathsf{COM}_{\mathsf{pCCA}}$. To this end, we first define a series of hybrids. Let H_i denote the following hybrid:

1. Interact with the adversary $\mathcal{A}$ on input $(1^\kappa, z)$ as in the trapdoor game. Initially, receive a message $(\texttt{start}, tag, v)$.
2. Create the shares $s_{m,n} \in \{0,1\}^{|v|}$ according to the protocol description of SSCOM. Continue the execution as in the trapdoor experiment, except with the following changes:
   a) Perform the commit phase of SSCOM as follows: For $m = 1, \dots, i$, commit to $s_{m,0}$ using $\mathsf{COM}_{\mathsf{pCCA}}$ for both the commitment with tag $(tag, m, 0)$ and tag $(tag, m, 1)$. For $m = i+1, \dots, \kappa$, commit to $s_{m,0}$ for the commitment with tag $(tag, m, 0)$ and to $s_{m,1}$ for the commitment with tag $(tag, m, 1)$, *i.e.* commit honestly.
   b) At the beginning of the unveil phase, extract the index vector I stored in the timed commitment using the `forced-open` algorithm.
   c) In the unveil phase, send $s_{m,n}$ for $m = 1, \dots, \kappa$ and $n \in \{0,1\}$. For $m \leq i$ and $n \in \{0,1\}$, send $s'_{m,I[m]} = s_{m,0}$ and $s'_{m,1-I[m]} = v \oplus s_{m,0}$, *i.e.* shares consistent with v and the commitments that will be opened.
3. Provide the pCCA oracle for $\mathcal{A}$ by forwarding messages between $\mathcal{A}$ and the pCCA oracle provided in the experiment.
4. Continue the internal execution of the trapdoor game and eventually output what $\mathcal{A}$ outputs.

By definition, H_0 is identical to an execution of the trapdoor game where $b = 0$, *i.e.* the commit and unveil phase are performed honestly. Similarly, H_κ is identical to an execution with $b = 1$, *i.e.* the commit and unveil phase are performed like the trapdoor algorithm would.

Suppose for the sake of contradiction that there exists an adversary that can distinguish between H_0 and H_κ with non-negligible probability, *i.e.* $|\Pr[\mathrm{out}_0 = 1] - \Pr[\mathrm{out}_\kappa = 1]| = \nu(\kappa)$ is non-negligible. Then, there exists an index $i = i(\kappa)$ such that the adversary can distinguish between H_i and H_{i+1} with non-negligible probability of at least $\nu(\kappa)/\kappa$. We show that this contradicts the pCCA hiding property of $\mathsf{COM}_{\mathsf{pCCA}}$.

Claim 3.1. *If $\mathsf{COM}_{\mathsf{pCCA}}$ is pCCA-secure, then out_i and out_{i+1} are (computationally) indistinguishable.*

Proof. Suppose for the sake of contradiction that for an index $i \in [\kappa]$, out_i and out_{i+1} are not computationally indistinguishable, *i.e.* $p = |\Pr[\mathrm{out}_i = 1] - \Pr[\mathrm{out}_{i+1} = 1]|$ is non-negligible. Then, we can construct an adversary $\mathcal{A}'$ against the pCCA-hiding property of $\mathsf{COM}_{\mathsf{pCCA}}$ with non-negligible advantage as follows:

On input $(1^\kappa, z)$, $\mathcal{A}'$ externally interacts with an instance of the pCCA hiding game. Internally, it runs an instance of the trapdoor game with adversary $\mathcal{A}$ on input $(1^\kappa, z)$ as follows:

1. Forward oracle queries by $\mathcal{A}$ to the pCCA oracle of the pCCA hiding game.

2. Sample $b' \xleftarrow{\$} \{0,1\}$ uniformly at random.
3. Play H_i, but perform the $\mathsf{COM}_{\mathsf{pCCA}}$ commitment with tag $(tag, i+1, b')$ as follows:
   - Send $(s_{i+1,0}, s_{i+1,1})$ and tag $(tag, i+1, b')$ to the pCCA hiding game as challenge.
   - Perform the commit phase for the commitment with tag $(tag, i+1, b')$ with the pCCA hiding game.
4. Continue the execution of H_i but abort the unveil phase if $I[i+1] = b'$. In this case, send b' to the hiding experiment.
5. Otherwise, continue the execution and output whatever $\mathcal{A}$ outputs.

If $\mathcal{A}'$ does not abort and the pCCA hiding game has the choice bit 0, *i.e.* commits to $s_{i+1,0}$, then $\mathcal{A}$'s view is distributed as in H_i. Otherwise, it is distributed as in H_{i+1}. Let p denote $\mathcal{A}$'s distinguishing advantage between out_i and out_{i+1}. By definition of $\mathcal{A}'$, its advantage in the pCCA hiding game is $p/2$, which is non-negligible if p is non-negligible, contradicting the pCCA hiding property of $\mathsf{COM}_{\mathsf{pCCA}}$. □

We can conclude that $|\Pr[\mathrm{out}_0 = 1] - \Pr[\mathrm{out}_\kappa = 1]| \leq 2\kappa \cdot \mathsf{negl}^{\mathsf{pCCA}}_{\mathsf{COM}_{\mathsf{pCCA}}}(\kappa)$ for a negligible function $\mathsf{negl}^{\mathsf{pCCA}}_{\mathsf{COM}_{\mathsf{pCCA}}}$ bounding an adversary's advantage in the pCCA hiding game for $\mathsf{COM}_{\mathsf{pCCA}}$. □

Theorem 3.2. *Let* SSCOM *be a trapdoor commitment scheme for the pCCA oracle of* $\mathsf{COM}_{\mathsf{pCCA}}$. *Let* TCOM *be a* $(T, \ell'(\ell(\kappa), \kappa))$*-weakly extractable timed commitment scheme for some polynomially bounded* $T > \ell'(\ell(\kappa), \kappa)$ *and sufficiently large* $\ell'(\ell(\kappa), \kappa)$. *Then,* SSCOM *is an* $\ell(\kappa)$*-simulation-sound commitment scheme.*

Proof of the Timed Simulation-Soundness of SSCOM.

In the following, we prove that SSCOM (Construction 1) is timed simulation-sound. The value committed to is determined as follows: Let $s^e_{0,0}, \ldots s^e_{\kappa,1} \in \{0,1\}^\kappa \cup \{\perp\}$ be the shares committed to using $\mathsf{COM}_{\mathsf{pCCA}}$. If there is a value $v \in \{0,1\}^\kappa$ encoded by a strict majority of the shares, we say that v is the value committed to. Otherwise, if no such value exists, we say that the value committed to is $\perp$.

Proof. We prove Theorem 3.2 using a hybrid argument. Let $q = q(\kappa)$ be a bound for the number of left commitment sessions in the timed simulation-soundness experiment.

Let H_i denote the execution of the simulation-soundness experiment where the first i left sessions are equivocated and the remaining sessions are honest, *i.e.* use the code of the honest committer C instead of $\mathsf{C}_{\mathrm{trap}}$. (For the honest sessions, the committer's private input is sampled at commit time.) H_0 refers to an execution where all left sessions are honest, H_q to an execution where all left sessions are equivocated, *i.e.* the standard execution of the simulation-soundness experiment.

Let $H_i^\star$ be identical to H_i, but stop the execution before the TCOM commitment to I in the single right session is unveiled. Let $S_{ext} = (s^e_{0,0}, s^e_{0,1}, \dots, s^e_{\kappa,0}, s^e_{\kappa,1})$ denote the list of shares committed to in the right session. As $\mathsf{COM}_{\mathsf{pCCA}}$ is perfectly binding, S_{ext} is well-defined. Let S denote the list of shares sent in the first-round message of the unveil phase. We say that S is *consistent* relative to I (which is well-defined as TCOM is perfectly binding) if the following holds:

1. S encodes a value $v' \in \{0,1\}^\kappa$ and S is locally consistent, *i.e.* $v' = s_{0,0} \oplus s_{0,1} = \dots = s_{\kappa,0} \oplus s_{\kappa,1}$ and
2. $s_{0,I[0]} = s^e_{0,I[0]} \wedge \dots \wedge s_{\kappa,I[\kappa]} = s^e_{\kappa,I[\kappa]}$, *i.e.* the shares are consistent with commitments that would be unveiled.

The adversary wins in $H_i^\star$ if S is consistent relative to I and reconstructs to a different value $v' \neq v$ such that $v' \in M$, where $v \in \{0,1\}$ is the value encoded by a strict majority of the shares in S_{ext} (if such a value exists) and $\perp$ otherwise.

We first show that in H_0, the execution where no commitments are equivocated, the adversary $\mathcal{A}$ is non-abusing, *i.e.* does not equivocate on the right side, except with negligible probability. Starting from H_0, we will show that this property is retained in subsequent hybrids. As a non-abusing adversary does not satisfy the winning condition (except with negligible probability), the claim follows.

Claim 3.2. *Let $T(\kappa) > \ell'(\ell(\kappa), \kappa)$ be a polynomial in κ. If TCOM is $\ell'(\ell(\kappa), \kappa)$-hiding for sufficiently large $\ell'(\ell(\kappa), \kappa)$ and perfectly binding and if $\mathsf{COM}_{\mathsf{pCCA}}$ is perfectly binding, then $\mathcal{A}$ is non-abusing in H_0.*

Proof. We prove Claim 3.2 by a reduction to the timed hiding property of TCOM. As we cannot simulate H_0 in the reduction to the hiding property, we instead show that $\mathcal{A}$ is non-abusing in $H_0^\star$. It then follows that $\mathcal{A}$ is also non-abusing in H_0.

Assume for the sake of contradiction that there exists an adversary $\mathcal{A}$ that is abusing in $H_0^\star$. First, we fix the coins of $\mathcal{A}$ and $H_0^\star$ up to the point before the TCOM commitment starts such that $\mathcal{A}$ has maximum success probability. We also fix $\mathcal{A}$'s advice z. Let $r_{\mathcal{A}}$ denote these coins of the adversary and r_H denote the coins of the hybrid. As the execution of $H_0^\star$ using these coins is deterministic until the TCOM commitment starts and $\mathsf{COM}_{\mathsf{pCCA}}$ is perfectly binding, the shares $s_{m,n}$ committed to in the right session are well-defined given the coins and the advice. As before, let S_{ext} denote the list containing these shares. We prove Claim 3.2 by a reduction to the non-uniform timed hiding property of TCOM. To this end, we construct an adversary $\mathcal{A}'$ against the hiding property of TCOM that is successful if $\mathcal{A}$ is abusing. $\mathcal{A}'$ works as follows:

1. On input $(1^\kappa, (r_{\mathcal{A}}, r_H, S_{ext}, z))$, internally start an instance of $H_0^\star$ using coins $r_{\mathcal{A}}$ for the adversary and r_H for $H_0^\star$. Externally, interact with the TCOM hiding experiment. Run the adversary $\mathcal{A}$ on input $(1^\kappa, z)$ and emulate $H_0^\star$ using r_H just before the TCOM commitment starts.
2. Sample $I \xleftarrow{\$} \{0,1\}^\kappa$ and send $(I, \overline{I})$ to the timed hiding experiment.

3. Perform the commit phase for TCOM in the right execution with the hiding experiment.

4. Receive the shares S in the unveil phase.

5. If $\mathcal{A}$ has sent a $(\mathtt{notify}, id, 1)$ message where id is the tag of the right commitment in either thread, output a random bit b.

6. If S is locally consistent with I and S_{ext}, output 0. If S is locally consistent with $\overline{I}$ and S_{ext}, output 1. Otherwise, output a random bit b.

In order to abuse, $\mathcal{A}$ must send shares in the unveil phase that are consistent with the index vector committed to with TCOM, which is either I or $\overline{I}$, as well as with the extracted shares. Thus, an abusing $\mathcal{A}$ can be used to distinguish between a commitment to I resp. $\overline{I}$. (It is easy to see that if $\mathcal{A}$ is abusing, S cannot be locally consistent with *both* I and $\overline{I}$.) If $\mathcal{A}$ is unsuccessful, *i.e.* the sent shares are inconsistent, we output a uniformly random bit.

It remains to show that $\mathcal{A}'$ is $\ell'(\ell(\kappa), \kappa)$-bounded. By assumption, $\mathcal{A}$ is legal. Thus, between the beginning of the timed commitment and the transmission of the shares $s_{m,n}$, the whole experiment, including the adversary $\mathcal{A}$, has spent at most ℓ steps, unless $\mathcal{A}$ has notified the receiver of a timeout. If ℓ' is sufficiently large to account for i) the overhead from internally emulating $H_0^\star$ and $\mathcal{A}$, ii) the computations performed by $\mathcal{A}'$ after the challenge commitment has been sent and iii) possible (polynomial) speed-up of the internally emulated $\mathcal{A}$ due to possible differences in the machine model between $\mathcal{A}'$ and $\mathcal{A}$, then $\mathcal{A}'$ is legal and its advantage in the hiding game is equal to the probability that $\mathcal{A}'$ is abusing, *i.e.* $\Pr[\mathrm{E}_{\mathrm{ABUSE}_0}] = \mathrm{Adv}^{\mathrm{Hiding}}_{\mathcal{A}',\mathsf{TCOM}}(\kappa)$, where $\mathrm{E}_{\mathrm{ABUSE}_0}$ denotes the event that the adversary is abusing in H_0.

This leads to a contradiction of the timed hiding property of TCOM if $\mathcal{A}$ is abusing with non-negligible probability. Thus, it holds that $\Pr[\mathrm{E}_{\mathrm{ABUSE}_0}] \leq \mathsf{negl}^{\mathrm{Hiding}}_{\mathsf{TCOM}}(\kappa)$, where $\mathsf{negl}^{\mathrm{Hiding}}_{\mathsf{TCOM}}$ is a negligible function denoting an adversary's advantage in the timed hiding game for TCOM.

By definition of $H_0^\star$ and H_0, $\mathcal{A}$ cannot win in H_0 if it cannot win in $H_0^\star$, as $\mathsf{COM}_{\mathsf{pCCA}}$ is perfectly binding. Thus, if $\mathcal{A}$ is non-abusing in $H_0^\star$, then it is also non-abusing in H_0 and the claim follows. □

Remark 3.3. In order to obtain a uniform reduction, changes to the proof and an additional assumption are necessary. With respect to the assumptions, the pCCA-secure commitment scheme has to be extractable by the reduction adversary, *e.g.* via rewinding, which is usually the case for non-malleable commitment schemes in the plain model. In particular, the scheme due to Goyal *et al.* [GRRV14] satisfies this property, even if the base commitments are not efficiently extractable. As the commit phases of $\mathsf{COM}_{\mathsf{pCCA}}$ have finished before the challenge phase of the reduction starts, rewinding does not interfere with the reduction.

Also, the timed hiding definition (Definition 2.9) has to be split in two phases. In a first phase before the actual commit phase starts, the steps performed by the polynomial-time adversary $\mathcal{A}$ are not counted. In this phase, the share commitments can be extracted, *e.g.* via rewinding, instead of providing their value via the non-uniform advice. In the second phase, the timed commitment is performed and the adversary's steps are counted as usual. As Definition 2.9 requires security against non-uniform adversaries, any timed commitment scheme secure according to this definition is also secure when considering the outlined variant of the timed hiding notion.

Lemma 3.1. *If $\mathcal{A}$ is non-abusing in H_0 and if* SSCOM *is a trapdoor commitment scheme for the pCCA oracle of* $\mathsf{COM}_{\mathsf{pCCA}}$ *and* $\mathsf{COM}_{\mathsf{pCCA}}$ *is a perfectly binding and pCCA-secure commitment scheme, then $\mathcal{A}$ is non-abusing in H_i for $i > 0$,* i.e. $\Pr[out_i = 1] \leq \mathsf{negl}(\kappa)$.

Proof. We prove Lemma 3.1 by induction. We have shown that $\mathcal{A}$ is non-abusing in H_0 and $H_0^\star$. We assume that $\mathcal{A}$ is non-abusing in H_{i-1} and $H_{i-1}^\star$ for some fixed $i \in [q(\kappa) - 1]$. Then, we can show that $\mathcal{A}$ is also non-abusing in H_i and $H_i^\star$ by a reduction to the trapdoor property of SSCOM for the pCCA oracle of $\mathsf{COM}_{\mathsf{pCCA}}$. To this end, we construct an adversary $\mathcal{A}'$ against the trapdoor property of SSCOM as follows:

1. On input $(1^\kappa, z)$, internally start an execution of $H_{i-1}^\star$.
2. In the i-th left session, sample $v_i \leftarrow \mathcal{D}_i$ and play the commitment with the trapdoor experiment on challenge value v_i.
3. Play the single right commitment honestly, but perform the share commitments with the pCCA oracle. After these commit phases have finished, obtain the $s_{m,n}^e$ from the pCCA oracle and determine the value committed to as follows: If there is a value $v \in \{0,1\}^\kappa$ encoded by a strict majority of the extracted shares, let v be the value committed to. Otherwise, set $v = \perp$. Then, continue the execution.
4. Upon receiving the $s'_{m,n}$ in the right session, check if they are locally consistent, and consistent with the index vector I and the extracted shares. If they reconstruct to a value $u \in \{0,1\}^\kappa$ such that $v \neq u$, output 1.
5. Otherwise, output 0.

Let p_{i-1} resp. p_i denote the probability that $\mathcal{A}$ is abusing in $H_{i-1}^\star$ resp. $H_i^\star$. The advantage of $\mathcal{A}'$ in the trapdoor game is $p = |p_i - p_{i-1}|$. If p is non-negligible, then $\mathcal{A}'$ wins the trapdoor game with non-negligible probability, contradicting the trapdoor property of SSCOM. It follows that the advantage of $\mathcal{A}'$ in $H_i^\star$ for $i \geq 1$ can be bounded by a negligible function $\mathsf{negl}_{\mathsf{SSCOM},\mathcal{O}}^{\mathrm{TD}}$ bounding an adversary's advantage against the trapdoor property of SSCOM, which is negligible by assumption. As $\Pr[\mathrm{out}_i = 1] \leq \Pr[\mathrm{out}_i^\star = 1]$, the claim follows. □

Let $\mathsf{negl}_{\mathsf{TCOM}}^{\mathrm{Hiding}}$ be a negligible function bounding a $\ell'(\ell(\kappa), \kappa)$-bounded adversary's advantage against the timed hiding property of TCOM. Then, it holds that $\mathrm{Adv}_{\mathcal{A},\mathsf{SSCOM},\ell}^{\mathrm{SIMSOUND}}(\kappa) \leq \mathsf{negl}_{\mathsf{TCOM}}^{\mathrm{Hiding}}(\kappa) + (q(\kappa) - 1) \cdot \mathsf{negl}_{\mathsf{SSCOM},\mathcal{O}}^{\mathrm{TD}}(\kappa)$, which is negligible. □

Looking ahead to the proof of our composable commitment scheme (Claim 3.4), we will need SSCOM commitments to be (not necessarily straight-line) extractable. The general setting is the following: The environment concurrently receives SSCOM commitments from the simulator. At the same time, it commits (in one session) in parallel using $\mathsf{COM}_{\mathsf{pCCA}}$ within SSCOM. In the reduction, the reduction adversary can play all commitments the environment receives as they are unrelated to the reduction. However, it has to extract the $\mathsf{COM}_{\mathsf{pCCA}}$ commitments performed in parallel by the environment in order to embed a challenge. As the "left sides" can be played honestly and can be rewound, we do not need the *robustness* guaranteed by some non-malleable commitments (*e.g.* [GRRV14]) where the left sides have to be simulated by the extractor without rewinding them. Thus, "non-robust" (parallel) extractability similar to the notion defined by Pass and Wee [PW09] is sufficient. We also recall that extractability for one commitment already implies parallel extractability in this setting [PW09]).

Corollary 3.1. *Let E' be an extractor for $\mathsf{COM}_{\mathsf{pCCA}}$ with overwhelming success probability (over the coins of the extractor and the adversary). Then, there exists an extractor E for SSCOM with overwhelming success probability (over the coins of the extractor and the adversary) with a runtime that is polynomial in the runtime of E'.*

Proof sketch. We prove Corollary 3.1 by constructing an extractor E for SSCOM as follows. Let E' denote the extractor for $\mathsf{COM}_{\mathsf{pCCA}}$.

- Internally execute E' in parallel and forward all messages related to $\mathsf{COM}_{\mathsf{pCCA}}$ instances where the adversary is the committer between E' and the adversary.
- Execute all instances of $\mathsf{COM}_{\mathsf{pCCA}}$ where the adversary is receiver honestly, also when rewinding.
- Emulate all other messages of SSCOM relative to the current state of the sessions resp. to the protocol description if no such state exists for the current session.
- If there is a value $v \in \{0,1\}^{\kappa}$ encoded by a strict majority of the shares, we say that v is the value committed to. Otherwise, if no such value exists, we say that the value committed to is $\perp$. (While we could require that *all* shares reconstruct to the same value, this would lead to *underextraction*: An honest receiver could accept an unveil to $v \neq \perp$ when, say, one unveiled share is different from the share committed to (which is not unveiled due to the choice of I), while the extractor would output $\perp$.)

The proof that a corrupted committer cannot open its commitment to a value different from the extracted one is implicit in the proof of Theorem 3.2. Also, it is easy to see that an adversary's view when interacting with the extractor is (computationally) indistinguishable from an interaction with an honest receiver. □

Depending on the setting, E may also use the code of the trapdoor committer for the left sides.

For example, the pCCA-secure commitment scheme due to Goyal *et al.* [GRRV14] admits such an extractor E'. In contrast to their construction in the context of non-malleability, our composite extractor can honestly answer challenges of concurrently executed commitments where the adversary acts as receiver. This is because here, the "left side" can be rewound during extraction.

Corollary 3.1 follows implicitly from the proof of Theorem 3.2.

Possible Instantiations.

Our construction SSCOM makes use of a weakly extractable timed commitment scheme TCOM as well as a pCCA-secure and perfectly binding commitment scheme $\mathsf{COM}_{\mathsf{pCCA}}$. A possible instantiation for the latter is the commitment scheme of Goyal *et al.* [GRRV14] which is pCCA-secure [BDH$^+$17], constant-round, non-black-box, extractable and perfectly binding if using *e.g.* the commitment scheme due to Blum [B81] based on one-way permutations as base commitment scheme. By instead using a perfectly binding and homomorphic commitment scheme[5], the construction becomes perfectly binding and black-box [BGR$^+$15; BDH$^+$17].

Corollary 3.2. *If constant-round, perfectly binding and homomorphic commitment schemes as well as constant-round weakly extractable timed commitment schemes with appropriate parameters exist, then* SSCOM *is a constant-round timed simulation-sound commitment scheme that makes black-box use of its building blocks only.*

An example for a constant-round homomorphic commitment scheme is the ElGamal commitment scheme based on the Decisional Diffie-Hellman (DDH) assumption [E84], which does not use non-black-box techniques. With respect to the timed commitment scheme, we can *e.g.* use the scheme due to Boneh and Naor [BN00] based on the generalized BBS assumption, which is constant-round and also does not use non-black-box techniques.

Corollary 3.3. *If the DDH assumption and the generalized BBS assumption hold for appropriate parameters, then there exists a constant-round, timed simulation-sound commitment scheme that makes use black-box use of its building blocks only.*

3.4. TLUC Security

The commitment scheme SSCOM with timed security constructed in Section 3.3 cannot be meaningfully used within UC-secure protocols due to the lack of a notion of time. In this section, we adapt UC security such that honest protocol parties can set up timers, very similar as *e.g.* in the definition of timed simulation-soundness (Definition 3.8). We

[5]Informally, a (non-interactive) commitment scheme COM with message space M is called *homomorphic* if commitments c, c' to values $v, v' \in M$ can be efficiently combined to a commitment c^* to the value $v \circ v' \in M$ for some operation $\circ$, *e.g.* addition or multiplication. Additionally, it is usually required that c^* is (computationally) indistinguishable from a fresh commitment to $v \circ v'$.

call the resulting notion *TLUC security*, short for “time-lock UC”, which allows to meaningfully use timed primitives within larger protocols.

As certain central definitions are easy to grasp at an intuitive level, but, at the same time, hard to formalize, we start with an informal introduction that conveys the necessary concepts and is sufficient to understand the following chapters.

3.4.1. TLUC Security in a Nutshell

Timed primitives such as timed commitment schemes can be meaningfully used in practice. Consider performing a coin-toss using a timed commitment scheme secure for, say, $t = 10^{15}$ steps. Assuming that the adversary can perform at most 10^{10} steps per second (equating 10 GHz, assuming that steps equate cycles)[6], a coin-toss using this timed commitment should be considered secure if the adversary’s second-round message comes within *e.g.* one second of receiving the timed commitment, with plenty time left as security margin.

TLUC Security. Unfortunately, this intuition is not easily captured in the UC framework, which neither offers a notion of time nor makes assumptions with respect to the (concrete) computational power of entities. Instead of considering a model with time or modifying the framework, we propose a variant of UC security, called TLUC security, that enables honest parties to check if more than ℓ steps have been performed since a certain point in the execution. This allows to capture the security guarantees of timed primitives and to use them in protocols.

With TLUC, parties can set up *timers* parameterized by an ID and a number of computation steps ℓ by sending $(\texttt{timer}, id, \ell)$ to the adversary[7]. At any point, a party that has set up a timer may check if it has expired, *i.e.* if the whole execution experiment has performed ℓ or more steps since the timer has been set up. This is done by sending $(\texttt{notify}, id)$ to the adversary. The adversary queries the environment if the timer has expired and answers with $(\texttt{notify}, id, b)$, where $b = 1$ denotes an expired timer and $b = 0$ an unexpired one.

Mechanisms. The correct handling of timers is ensured by considering only *legal environments* and *legal adversaries*. Intuitively, legal environments correctly account for timers set up by honest parties by never under-estimating the number of computation steps performed by the execution experiment relative to a presumptive execution of a protocol π (counting obliviously of the parties’ inputs and outputs) and adversary $\mathcal{A}$, denoted by $\mathcal{Z}[\pi, \mathcal{A}]$. This guarantees that timed assumptions protect against environment and adversary, but can be broken by the simulator in polynomial time without triggering a timeout (as the environment $\mathcal{Z}[\pi, \mathcal{A}]$ always counts relative to π and $\mathcal{A}$, even when

[6]This is even more plausible when using cryptographic assumptions that are belived to be hard even for parallel adversaries.

[7]In contrast to stand-alone experiments where `timer` messages are not parameterlized with the timed security parameter, we have chosen to do so in the TLUC setting because the mechanism should be agnostic of the currently executed protocol and its timed security parameter.

interacting with ϕ and $\mathcal{S}$). For technical reasons, we require handling of timers and inquiries to go through the adversary. An adversary is legal if it immediately and correctly forwards timer setup messages or status inquiries by honest parties, as well as the environment's responses. Based on this, we define TLUC emulation as a special case of UC emulation, and consider legal adversaries and environments only. At first glance, this might seem restrictive, but when considering standard UC protocols without timers, it is easy to see that the notions are equivalent for them. Thus, the restrictions essentially only apply when protocols with timers are executed.

Properties of TLUC Security. As we consider only a subset of UC environments and adversaries, properties of UC security do not necessarily carry over to TLUC security, at least for protocols using timers. To the contrary, even properties such as the completeness of the dummy adversary are difficult to prove if concrete time bounds must be adhered to. We show several properties such as transitivity with UC protocols, *i.e.* protocols whose security does not rely on timers[8], completeness of the dummy adversary or full compatibility with UC security as well as UC reusability, meaning that all UC-secure protocols are also TLUC-secure and can be composed with TLUC protocols without loss of security. With respect to the latter, we state the single instance composition theorem.

The ability of the simulator to break timed assumptions while environment and real-world adversary are unable to do so unnoticed is sufficient to construct a composable commitment scheme in the plain model. When, *e.g.* combining our commitment scheme with a UC-secure general MPC protocol in the $\mathcal{F}_{\mathrm{COM}}$- or $\mathcal{F}_{\mathrm{MCOM}}$-hybrid model[9], we obtain a TLUC-secure composable general MPC protocol in the plain model.

While composable MPC in the plain model is already possible in a number of other frameworks, previous approaches rely on some sort of super-polynomial or non-uniform simulation. The first may affect the security of concurrently executed protocols relying on polynomial-time hardness assumptions, resulting in limited *environmental friendliness* as defined by [CLP13a] or limited UC reusability. TLUC security only considers entities that run in strict polynomial time. The second may affect the security of protocols that have been previously started, even ones that are secure against non-uniform adversaries. Our feasibility results hold for uniform adversaries, simulators and environments.

Thus, TLUC security is the first notion that features composable constant-round black-box MPC in the plain model from standard (timed) assumptions, full UC reusability as well as full environmental friendliness and does not hurt the security of previously started protocols relying on polynomial-time assumptions.

We continue with an (almost) complete and formal treatment of TLUC security. The remaining (technical) parts can be found Appendix A.1.

[8]A UC protocol π that UC-realizes an ideal functionality $\mathcal{F}$ may of course send `timer` messages. However, as UC emulation also considers adversaries and environments that handle these messages arbitrarily, the security of π cannot rely on them.

[9]$\mathcal{F}_{\mathrm{MCOM}}$ and the multi-session extension $\widehat{\mathcal{F}}_{\mathrm{COM}}$ of $\mathcal{F}_{\mathrm{COM}}$ are equivalent [CR03].

3.4.2. Protocol Emulation

We start with the definition of TLUC emulation, which is defined in analogy to UC emulation.

Definition 3.9 (TLUC Emulation)**.** Let π and ϕ be probabilistic polynomial-time (PPT) protocols. We say that π TLUC-emulates ϕ if for every legal PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for every legal PPT environment $\mathcal{Z}[\pi, \mathcal{A}]$, there exists a negligible function negl such that for every $\kappa \in \mathbb{N}$ and every $z \in \{0,1\}^*$ it holds that

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

If π TLUC-emulates ϕ, we write $\pi \geq_{\text{TLUC}} \phi$. When omitting the non-uniform input z, the notion of protocol emulation is uniform.

Remark 3.4. In Definition 3.9, the environment $\mathcal{Z}$ counts the steps according to the execution with π and $\mathcal{A}$ even if it actually interacts with ϕ and $\mathcal{S}$. This allows the PPT-bounded simulator $\mathcal{S}$ to perform more steps than the adversary $\mathcal{A}$ without triggering a timeout, allowing it to break timed assumptions unnoticed. If ϕ is a UC protocol, its security is not affected by such a powerful simulator. In contrast, if ϕ is a protocol uses timers, honest parties of the protocol ϕ may not rely on timing assumptions as the adversary $\mathcal{S}$ is allowed to violate them unnoticed.

Meaningfulness of TLUC Security. When introducing a new security notion, it is important to argue that it does not allow to prove the security of "obviously insecure" protocols.

The basic idea behind TLUC security is the very same as behind established simulation-based security notions, where a protocol's security is defined through the ideal functionality it realizes (together with the rules of the execution). For simulation-based security notions, care has to be taken that the simulator's capabilities do not affect the security guarantees of the ideal functionality. For example, SPS security is not meaningful for ideal functionalities that use a polynomial-time hardness assumption like a signature scheme that can be broken by the super-polynomial simulator. As TLUC simulations are always polynomial-time, they do not affect an ideal functionality that makes use of polynomial-time assumptions. To this end, we had to take care that the simulator cannot learn *e.g.* an honest party's input through "timing side-channels" resulting from timers it may set up. For a discussion, see Appendix A.1.2.

In total analogy to both UC security and other composable security notions that admit general MPC in the plain model, we can show a strong impossibility result. This underlines that the new mechanism of timers, if implemented correctly, does not help the simulator *per se*.

Plausibility of TLUC Simulation. Canetti *et al.* [CDPW07] have raised the question of *simulation plausibility* with respect to different simulation strategies and deniability.

They state that "if the resources required to simulate a protocol session are readily available, then we say the protocol session is *plausibly deniable* (since it is plausible that information obtained from the protocol was the result of a simulation). If the resources required to simulate are difficult or impossible to obtain, then there is no guarantee of plausible deniability (since it will be difficult to convince others that an incriminating protocol transcript was the result of a simulation)" [CDPW07]. In standard UC, the existence of a simulation strategy is "plausible" as in the presence of a trusted setup, the task of simulation does not asymptotically require any computational power that other entities are not capable of. In particular, protocol parties can provide "fake" transcripts of interactions that cannot be verified by other parties that do not have access to the setup used in the transcript. (Still, UC security does not rule out that a simulator has to perform an "efficient" computation requiring $\Theta(\kappa^{100})$ steps, while all other entities have runtime in *e.g.* $\Theta(\kappa)$ with a small constant factor only.)

In contrast, when considering *e.g.* SPS security, the simulator is required to have computational power (*i.e. online* super-polynomial computation capabilities) that is not believed to exist and which are not admitted to other entities such as protocol parties or the environment. In this setting, a protocol transcript may be meaningful for other parties that have not participated in the protocol execution in question: Either, the transcript was the result of a real execution or its creation required super-polynomial resources, which are not believed to be available. Thus, a protocol participation cannot be plausibly denied.[10]

In advanced settings such as the Angel-based security framework, the question of simulation plausibility is harder to answer as the power available to the simulator is less clear-cut. (Moreover, a complexity advantage can often be traded for the ability to rewind, *e.g.* in [CLP10; CLP13a].) Indeed, as Canetti *et al.* [CDPW07] argue, if the Imaginary Angel of [PS04] "were to somehow be made practical in the real world, all security would be lost", making the simulation implausible. Still, Prabhakaran and Sahai [PS04] have argued the plausibility of their Imaginary Angel by showing that it can be realized in the $\mathcal{F}_{\text{CRS}}$-hybrid model from one-way functions. We show a similar result for our composable commitment scheme in Section 3.6.2.

In TLUC, we believe that the plausibility of the simulation depends largely on the size of the (only temporary) gap between simulator and environment. If this gap is very large, *i.e.* if the simulator has to perform many more computation steps than the environment is allowed to while a timer is active, simulation may be less plausible than in a setting where this gap is very small, consisting only of very few steps. To some extent, this can be adjusted via a protocol's parameters and the underlying timed assumption. However, the asymmetry can always be "caught up" by a polynomial-time protocol party—allowing it to fake a transcript given "enough time" to internally execute the simulator. In any case, TLUC simulators are asymptotically as efficient as UC

[10]This argument assumes that a simulated *transcript* cannot be obtained in an efficient manner. Depending on the protocol, this may or may not be the case. For example, there may be protocols where straight-line simulation provably require super-polynomial resources, while merely simulating a transcript can be efficiently done using rewinding.

simulators and their existence would *never* endanger other polynomial-time hardness assumptions.

3.4.3. Properties of TLUC Security

Having defined protocol emulation, we can state important properties of TLUC security in analogy to properties of UC security.

Proposition 3.2 (Legality of the Dummy Adversary)**.** *The dummy adversary $\mathcal{D}$ is legal.*

Proposition 3.2 immediately follows from the definition of the dummy adversary in the UC framework.

As in UC security, it is sufficient to show protocol emulation with respect to the dummy adversary.

Proposition 3.3 (Completeness of the Dummy Adversary)**.** *Let π and ϕ be PPT protocols. Then, $\pi \geq_{\text{TLUC}} \phi$ if and only if π TLUC-emulates ϕ with respect to the dummy adversary.*

Proof. Clearly, if π TLUC-emulates ϕ, then π TLUC-emulates ϕ with respect to the dummy adversary. We now show the converse. Let π TLUC-emulate ϕ with respect to the dummy adversary, *i.e.* there exists a simulator $\mathcal{S}_{\mathcal{D}}$ such that for every legal PPT environment $\mathcal{Z}[\pi, \mathcal{D}]$, it holds that

$$\mathsf{Exec}(\pi, \mathcal{D}, \mathcal{Z}[\pi, \mathcal{D}]) \approx \mathsf{Exec}(\phi, \mathcal{S}_{\mathcal{D}}, \mathcal{Z}[\pi, \mathcal{D}]) \tag{3.1}$$

Consider the type-1 routing environment[11] $\mathcal{Z}_{\mathcal{D}}$ that expects to interact with the dummy adversary $\mathcal{D}$ and protocol π as the environment that internally runs the legal environment $\mathcal{Z}[\pi, \mathcal{A}]$ and the legal adversary $\mathcal{A}$ and relays all messages between (its internal simulation of) $\mathcal{A}$ and the dummy adversary $\mathcal{D}$ as well as (its internal simulation of) $\mathcal{Z}[\pi, \mathcal{A}]$ and the challenge protocol π. Eventually, $\mathcal{Z}_{\mathcal{D}}$ outputs what $\mathcal{Z}$ outputs.

By Proposition A.1, $\mathcal{Z}_{\mathcal{D}}$ is legal. Also, the views of $\mathcal{Z}$ and $\mathcal{A}$ remain identically distributed and timers set up in $\mathsf{Exec}(\pi, \mathcal{A}, \mathcal{Z}[\pi, \mathcal{A}])$ are handled identically to timers set up in $\mathsf{Exec}(\pi, \mathcal{D}, \mathcal{Z}_{\mathcal{D}}[\pi, \mathcal{D}])$ and vice versa.

It follows that

$$\mathsf{Exec}(\pi, \mathcal{A}, \mathcal{Z}[\pi, \mathcal{A}]) \equiv \mathsf{Exec}(\pi, \mathcal{D}, \mathcal{Z}_{\mathcal{D}}[\pi, \mathcal{D}]). \tag{3.2}$$

[11]Routing environments are special classes of environments defined in Definition A.3 that internally execute another environment as well as a protocol (type-2) or adversary (type-1). Informally, they are called routing environments because they only route messages between the internally emulated machines and the challenge protocol and adversary. Clearly, the number of steps performed by a routing environment may be differerent than the number of steps calculated by its internally executed environment, *e.g.* due to emulation overhead, a different adversary or a different protocol. This seemingly requires a legal routing environment to handle timers differently. As this would lead to changes in the internally emulated environment's view, this is undesirable. However, we can show that for the cases we are interested in, the routing environment can leave the handling of timers to its internally emulated environment.

Now, consider the execution where $\mathcal{Z}_\mathcal{D}$ interacts with the simulator for the dummy adversary $\mathcal{S}_\mathcal{D}$ and protocol ϕ. Using Equation (3.1) and the fact that $\mathcal{Z}_\mathcal{D}$ is legal, it follows that $\mathsf{Exec}(\pi, \mathcal{D}, \mathcal{Z}_\mathcal{D}[\pi, \mathcal{D}])$ and $\mathsf{Exec}(\phi, \mathcal{S}_\mathcal{D}, \mathcal{Z}_\mathcal{D}[\pi, \mathcal{D}])$ are (computationally) indistinguishable. We now construct the simulator $\mathcal{S}$ for $\mathcal{A}$ and π from $\mathcal{S}_\mathcal{D}$: $\mathcal{S}$ internally runs instances of $\mathcal{A}$ and $\mathcal{S}_\mathcal{D}$ as follows:

- Messages from ϕ to the adversary are forwarded to $\mathcal{S}_\mathcal{D}$.
- Messages from $\mathcal{S}_\mathcal{D}$ to ϕ are forwarded to ϕ.
- Messages from $\mathcal{S}_\mathcal{D}$ to the environment are forwarded to $\mathcal{A}$.
- Messages from $\mathcal{A}$ to π are forwarded to $\mathcal{S}_\mathcal{D}$.
- Messages from $\mathcal{A}$ to the environment are forwarded to the environment.
- Messages from the environment to the adversary are forwarded to $\mathcal{A}$.

By applying Proposition A.1 again, it follows that

$$\mathsf{Exec}(\phi, \mathcal{S}_\mathcal{D}, \mathcal{Z}_\mathcal{D}[\pi, \mathcal{D}]) \equiv \mathsf{Exec}(\phi, \mathcal{S}, \mathcal{Z}[\pi, \mathcal{A}]) \tag{3.3}$$

Combining Equations (3.1) to (3.3), the claim follows. □

TLUC security is also compatible with UC security, meaning that UC-secure protocols are also TLUC-secure.

Proposition 3.4 (Compatibility with UC Security). *Let π, ϕ be PPT protocols such that $\pi \geq_{\text{UC}} \phi$. Then, $\pi \geq_{\text{TLUC}} \phi$.*

Proof. As TLUC emulation is a special case of UC emulation and the class of simulators is not restricted, Proposition 3.4 trivially follows. □

Transitivity. In contrast to UC security, TLUC security is not transitive.

Proposition 3.5. *There exist PPT protocols π_1, π_2, π_3 such that $\pi_1 \geq_{\text{TLUC}} \pi_2$ and $\pi_2 \geq_{\text{TLUC}} \pi_3$ but $\pi_1 \not\geq_{\text{TLUC}} \pi_3$.*

Proof. Let π_3 the ideal protocol of the commitment functionality $\mathcal{F}_{\text{COM}}$. Let π_2 be the protocol π_{MCOM} (with slight syntactical modifications towards realizing $\mathcal{F}_{\text{COM}}$ instead of $\mathcal{F}_{\text{MCOM}}$) from Section 3.5. Let π_1 be the following protocol:

- Upon receiving $(\texttt{commit}, \mathit{sid}, b)$ as input, the committer samples $b' \overset{\$}{\leftarrow} \{0,1\}$ and sends $c = b \oplus b'$ to the receiver, which stores c and outputs $(\texttt{committed}, \mathit{sid})$.
- Upon receiving $(\texttt{unveil}, \mathit{sid})$ as input, the committer sends (b, b') to the receiver. If $c = b \oplus b'$, the receiver outputs $(\texttt{unveil}, \mathit{sid}, b)$.

Clearly, π_1 does not realize $\mathcal{F}_{\text{COM}}$, as the commitment is not extractable (and not even binding for a malicious committer). However, one can show that $\pi_1 \geq_{\text{TLUC}} \pi_2$ because, informally, π_2 gives no security guarantees in this setting, *cf.* Remark 3.4. Roughly, the simulator $\mathcal{S}$ for π_1 acts as follows: If the sender is corrupted, the simulator must be able to equivocate the commitment in π_2. This is possible as the simulator is able to determine the output of the coin-toss for the equivocation CRS. (Here, the corrupted committer is played by the simulator and not the environment.) Conversely, if the receiver is corrupted, the simulator sends a random value $c \overset{\$}{\leftarrow} \{0,1\}$ in the simulation of π_1. Later on, when it learns the committed bit b in π_2, the simulator sends $(b, c \oplus b)$ as unveil message. □

Note that the above simulator is not a legal adversary. While this is not necessary to argue that $\pi_1 \geq_{\text{TLUC}} \pi_2$, the prerequisite $\pi_2 \geq_{\text{TLUC}} \pi_3$ only requires the existence of a simulator for a *legal* adversary for π_1, which $\mathcal{S}$ is not.

However, we can state the following weaker and useful properties.

Corollary 3.4 (Transitivity for UC Protocols). *Let π_1, π_2, π_3 be PPT protocols. If $\pi_1 \geq_{\text{UC}} \pi_2$ and $\pi_2 \geq_{\text{UC}} \pi_3$, then it holds that $\pi_1 \geq_{\text{TLUC}} \pi_3$.*

Also, TLUC emulation is transitive in conjunction with UC emulation.

Proposition 3.6 (TLUC-UC Transitivity). *Let π_1, π_2, π_3 be PPT protocols. If $\pi_1 \geq_{\text{TLUC}} \pi_2$ and $\pi_2 \geq_{\text{UC}} \pi_3$, then it holds that $\pi_1 \geq_{\text{TLUC}} \pi_3$.*

Proof sketch. Let $\mathcal{A}$ be a legal adversary. Since $\pi_1 \geq_{\text{TLUC}} \pi_2$, there exists a (possibly not legal) PPT TLUC simulator $\mathcal{S}_1$ for every legal PPT adversary $\mathcal{A}$ such that for every legal PPT environment $\mathcal{Z}[\pi_1, \mathcal{A}]$, it holds that

$$\mathsf{Exec}(\pi_1, \mathcal{A}, \mathcal{Z}[\pi_1, \mathcal{A}]) \approx \mathsf{Exec}(\pi_2, \mathcal{S}_1, \mathcal{Z}[\pi_1, \mathcal{A}]) \tag{3.4}$$

and since $\pi_2 \geq_{\text{UC}} \pi_3$, there exists a PPT simulator $\mathcal{S}_2$ for every (not necessarily legal) PPT adversary $\mathcal{B}$ such that

$$\mathsf{Exec}(\pi_2, \mathcal{B}, \mathcal{Z}) \approx \mathsf{Exec}(\pi_3, \mathcal{S}_2, \mathcal{Z}) \tag{3.5}$$

As Equation (3.5) quantifies over all PPT adversaries and PPT environments (including legal ones), it implies the existence of a simulator for the PPT adversary / simulator $\mathcal{S}_1$. Combining Equations (3.4) and (3.5), the claim follows. □

Composition. In the following, we consider the case of a protocol ρ that makes one subroutine call to a protocol ϕ.

Theorem 3.3 (Single Instance Composition Theorem). *Let π, ϕ be subroutine-respecting PPT protocols such that $\pi \geq_{\text{TLUC}} \phi$. Let ρ be a PPT protocol that makes one subroutine call to ϕ. Then, $\rho^{\phi \to \pi} \geq_{\text{TLUC}} \rho$.*

Proof. In the following, we show that if π TLUC-emulates ϕ for the dummy adversary, then $\rho^{\phi\to\pi}$ emulates ρ for the dummy adversary, *i.e.* for every legal PPT environment, there exists a PPT simulator $\mathcal{S}_\rho$ such that

$$\mathsf{Exec}(\rho^{\phi\to\pi}, \mathcal{D}, \mathcal{Z}[\rho^{\phi\to\pi}, \mathcal{D}]) \approx \mathsf{Exec}(\rho, \mathcal{S}_\rho, \mathcal{Z}[\rho^{\phi\to\pi}, \mathcal{D}]). \tag{3.6}$$

By Proposition 3.3, this is sufficient and implies the general case. First, we construct a simulator $\mathcal{S}_\rho$ for $\rho^{\phi\to\pi}$. Internally, $\mathcal{S}_\rho$ runs the simulator $\mathcal{S}_\mathcal{D}$ for π and handles messages as follows:

- Messages from protocol parties of ρ are sent to the environment.
- Messages from the environment to the protocol parties of ρ are sent to these parties.
- Messages from protocol parties of ϕ are sent to $\mathcal{S}_\mathcal{D}$.
- Messages from $\mathcal{S}_\mathcal{D}$ to ϕ are sent to ϕ.
- Messages from the environment to the protocol parties of π are sent to $\mathcal{S}_\mathcal{D}$.
- Messages from $\mathcal{S}_\mathcal{D}$ to the environment are sent to the environment.

We show that $\mathcal{S}_\rho$ is a valid simulator for $\rho^{\phi\to\pi}$ and $\mathcal{D}$ by contradiction: Suppose there exists an environment $\mathcal{Z}[\rho^{\phi\to\pi}, \mathcal{D}]$ that can distinguish between the execution with $\rho^{\phi\to\pi}$ and $\mathcal{D}$ and the execution with ρ and $\mathcal{S}_\rho$, *i.e.*

$$\mathsf{Exec}(\rho^{\phi\to\pi}, \mathcal{D}, \mathcal{Z}[\rho^{\phi\to\pi}, \mathcal{D}]) \not\approx \mathsf{Exec}(\rho, \mathcal{S}_\rho, \mathcal{Z}[\rho^{\phi\to\pi}, \mathcal{D}]). \tag{3.7}$$

Let $\mathcal{Z}'[\pi, \mathcal{D}]$ be the type-2 routing environment (*cf.* Definition A.3) that internally runs $\mathcal{Z}[\rho^{\phi\to\pi}, \mathcal{D}]$ and ρ.

Using Proposition A.1 and the definition of $\mathcal{S}_\rho$, it is easy to see that $\mathcal{Z}$'s view when emulated by $\mathcal{Z}'$ is identically distributed as in the execution of $\rho^{\phi\to\pi}$ and $\mathcal{A}$ resp. the execution of ρ and $\mathcal{S}_\mathcal{D}$, depending on the challenge protocol of $\mathcal{Z}'$. By Proposition A.1, $\mathcal{Z}'$ is legal for π and $\mathcal{D}$ if $\mathcal{Z}$ is legal for $\rho^{\phi\to\pi}$ and $\mathcal{D}$.

Thus, the distinguishing advantage of $\mathcal{Z}'$ for π and ϕ is identical to that of $\mathcal{Z}$ for $\rho^{\phi\to\pi}$ and ρ and Equation (3.7) implies that

$$\mathsf{Exec}(\pi, \mathcal{D}, \mathcal{Z}'[\pi, \mathcal{D}]) \not\approx \mathsf{Exec}(\phi, \mathcal{S}_\mathcal{D}, \mathcal{Z}'[\pi, \mathcal{D}]), \tag{3.8}$$

contradicting the assumption that $\pi \geq_{\text{TLUC}} \phi$. Overall, it follows that if $\pi \geq_{\text{TLUC}} \phi$, then $\rho^{\phi\to\pi} \geq_{\text{TLUC}} \rho$. □

Let ρ be a PPT protocol that UC-emulates the ideal protocol IDEAL($\mathcal{G}$) of some ideal functionality $\mathcal{G}$ and makes one subroutine call to the ideal protocol IDEAL($\mathcal{F}$) of some ideal functionality $\mathcal{F}$. Using Propositions 3.4 and 3.6 and Theorem 3.3, we can import ρ into TLUC, replace IDEAL($\mathcal{F}$) with an appropriate TLUC protocol while preserving security and conclude that the resulting composite protocol TLUC-emulates IDEAL($\mathcal{G}$).

Corollary 3.5 (UC Reusability)**.** *Let π and ϕ be subroutine-respecting PPT protocols such that $\pi \geq_{\mathrm{TLUC}} \phi$. Let ρ be a PPT protocol that makes one subroutine call to ϕ such that $\rho \geq_{\mathrm{UC}} \sigma$. Then, $\rho^{\phi \to \pi} \geq_{\mathrm{TLUC}} \sigma$.*

Unfortunately, TLUC security is not closed under universal composition. This means that there exist subroutine-respecting protocols π and ϕ such that $\pi \geq_{\mathrm{TLUC}} \phi$ holds, but $\rho^{\phi \to \pi}$ does not TLUC-emulate ρ, where ρ makes *multiple* subroutine calls to ϕ.

Proposition 3.7. *Let π, ϕ be subroutine-respecting PPT protocols such that $\pi \geq_{\mathrm{TLUC}} \phi$. There exists a PPT protocol ρ that makes multiple subroutine calls to ϕ such that $\rho^{\phi \to \pi} \not\geq_{\mathrm{TLUC}} \rho$.*

As an example, take the commitment protocol π_{MCOM} (*cf.* Section 3.5) and replace the pCCA-secure commitment scheme $\mathsf{COM}_{\mathsf{CCA}}$ used in SSCOM with a malleable extractable commitment scheme. One can easily prove that this protocol still TLUC-realizes a single instance of $\mathcal{F}_{\mathrm{COM}}$, but is not even concurrently self-composable.

If π UC-emulates ϕ, we recover a UC-like composition theorem, which does not place a bound on the number of subroutine calls by ρ to ϕ.

Proposition 3.8 (Composition Theorem for UC Protocols)**.** *Let π, ϕ be subroutine-respecting PPT protocols such that $\pi \geq_{\mathrm{UC}} \phi$. Let ρ be a PPT protocol. Then, $\rho^{\phi \to \pi} \geq_{\mathrm{TLUC}} \rho$.*

Proof. Using the UC composition theorem, we obtain that $\rho^{\phi \to \pi} \geq_{\mathrm{UC}} \rho$. Using that TLUC security is compatible with TLUC security (Proposition 3.4), it follows that $\rho^{ptp} \geq_{\mathrm{UC}} \rho$. □

Environmental Friendliness.

Being a special case of UC security, TLUC security fulfills the important property of environmental friendliness (*cf.* Section 2.5).

Proposition 3.9 (Environmental Friendliness of TLUC Security)**.** *Let π be a PPT protocol that TLUC-emulates the ideal protocol of some functionality $\mathcal{G}$. Then, π is friendly to every (non-timed) game-based property P of a protocol Π with property P.*

Proof. UC security is friendly to all (non-timed) game-based properties [CLP13a]. As TLUC security is a special case of UC security (and, in particular, all entities run in polynomial time), the claim follows. □

Protocols running alongside composable MPC protocols may not only be affected by super-polynomial simulation, but also by non-uniform simulation. For example, Lin, Pass, and Venkitasubramaniam [LPV09] propose a variant of UC security where the environment runs in uniform polynomial time, while the simulator runs in non-uniform polynomial time. The non-uniform input of the simulator may impact the security of protocols that have started before the input is given to the simulator—even if these protocols are secure against non-uniform adversaries. As the definition of environmental friendliness is non-uniform, it does not capture this property.

The simulation for our composable commitment scheme (Section 3.5) is uniform. Our constructions thus do not adversely affect security properties of previously started protocols that hold against polynomial-time adversaries.

Remark 3.5. Environmental friendliness as defined by [CLP13a] is not meaningful for *timed* game-based properties such as the timed hiding property of a timed commitment scheme.

When considering an ideal functionality $\mathcal{F}$ and a concurrently executed protocol π using timed assumptions, the functionality $\mathcal{F}$ may already be unfriendly to timed properties of π. For example, $\mathcal{F}$ may perform computations that break time-lock puzzles used in π.

In the experiment of environmental friendliness, no simulator is used. The (presumptive) simulator is only used to show that a protocol π is as friendly as a functionality $\mathcal{F}$ (which may already be unfriendly in our setting). Thus, the problems with respect to environmental friendliness for timed security properties start well before considering the effects of the simulation, which may have additional negative effects.

To the best of our knowledge, this outlined variant of environmental friendliness for timed game-based properties is not fulfilled by *any* security notion for composable MPC—not even by UC security.

Non-Triviality. While there exists no general and formal definition of *non-triviality* in the UC framework, Canetti *et al.* [CLOS02] consider a protocol π to be a non-trivial realization of $\mathcal{F}$ if $\pi \geq_{\mathrm{UC}} \mathrm{IDEAL}(\mathcal{F})$ and for every adversary $\mathcal{A}$ that deliver all messages and does not corrupt any party, the simulator $\mathcal{S}$ allows all outputs generated by $\mathcal{F}$.

With TLUC security, this notion is not sufficient as it does not consider the possibility that a protocol execution aborts due to timeouts, which may, depending *e.g.* on the environment, occur even if the adversary delivers all messages.

As an example, let π be a protocol that non-trivially UC-emulates $\mathcal{F}_{\mathrm{COM}}$ and takes $t(\kappa)$ steps to execute successfully if all parties are honest. Now, let π' be the protocol that is identical to π, with the following exception. When receiving its input, the honest committer sets up a timer with $10t(\kappa)$ steps. At the onset of the unveil phase, it checks if the timer has expired and halts upon expiration. Clearly, π' should be considered non-trivial.

However, there exists a legal environment such that π' never generates output even if the legal adversary delivers all messages. As we do not want π' to be considered trivial if there also exists a legal environment $\mathcal{Z}$ for which π' always generates an output under the conditions outlined in [CLOS02], we adapt the notion of [CLOS02] to account for this[12].

Note that non-triviality may be lost under composition. To this end, take a protocol ρ that makes one subroutine call to some protocol ϕ and is non-trivial. Replacing ϕ in ρ with its realization π that takes more steps than ϕ may make the composed

[12]Legal environments count the number of computation steps independent of the parties' inputs. If this is done in an appropriate way, the existence of *one* environment that provides fixed input implies the existence of *other* environments with all possible inputs such that π always generates an output.

protocol $\rho^{\phi \to \pi}$ trivial as timers in ρ may *always* be triggered due to the additional steps performed by the protocol π. However, that this does *not* render $\rho^{\phi \to \pi}$ insecure.

Impossibility Results. The well-known impossibility results due to Canetti and Fischlin [CF01] state that there is no bilateral (*i.e.* involving two communicating parties) and terminating (in the sense of an output for the honest receiver with a certain probability) protocol π that UC-realizes $\mathcal{F}_{\text{COM}}$ in the plain model. This is due to the fact that if a protocol π is in the plain model, an environment is able to internally emulate every (presumptive) UC simulator for π.

We state the following variant of the impossibility result of [CF01] for TLUC-realizing $\mathcal{F}_{\text{COM}}$ in the plain model:

Theorem 3.4. *There exists no bilateral, non-trivial protocol π in the plain model where at most one party sets up timers such that π TLUC-realizes $\mathcal{F}_{\text{COM}}$.*

Proof. The case where no party sets up timers is easily recovered from the proof of Canetti and Fischlin [CF01].

We now consider the case that exactly one party sets up timers. The following proof follows the same outline as the proof of the impossibility result in [CF01]. Suppose that $\pi \geq_{\text{TLUC}} \text{IDEAL}(\mathcal{F}_{\text{COM}})$, *i.e.* for every legal PPT adversary, there exists a PPT simulator $\mathcal{S}$ for every legal PPT environment. We first consider the case where the committer is the party that sets up the timer(s). We use the presumptive simulator $\mathcal{S}$ to construct a legal environment $\mathcal{Z}[\pi, \mathcal{D}]$ expecting to interact with the dummy adversary for which there is no simulator, contradicting the assumption.

The environment $\mathcal{Z}[\pi, \mathcal{D}]$ works as follows: Initially, it orders the adversary to corrupt the committer and internally executes the simulator $\mathcal{S}$ for the corrupted *receiver* to generate an equivocal commitment. Prior to the beginning of the unveil phase, it samples a bit $b \xleftarrow{\$} \{0,1\}$ and instructs the simulator to decommit to b. `timer` and `notify` messages for resp. from the internal instance of $\mathcal{S}$ are handled by $\mathcal{Z}$ relative to a real execution where the committer is honest. This is possible without b as legal environments handle these messages independent of parties' inputs and outputs.

As there are no "real" timers if the committer is corrupted and π is non-trivial, the execution of π will terminate. Furthermore, $\mathcal{Z}[\pi, \mathcal{D}]$ is a legal environment.

If the environment is in the real execution of the protocol π, then the internally emulated simulator can, due to the correctness of the simulation, unveil the commitment to the bit b, except with negligible probability. In particular, the `timer` and `notify` messages are handled as in an interaction with a legal adversary relative to the internally emulated instance of $\mathcal{S}$.

If the environment is in the ideal execution of $\mathcal{F}_{\text{COM}}$, the whole execution is independent of b until the onset of the unveil phase. Thus, *any* simulator will extract the wrong bit b' with probability $1/2$, where the probability is over the coins of the simulator and the environment. Even if the internally emulated simulator failed during the unveil phase with non-negligible probability, this would allow the environment to distinguish as such a failure does not occur in the real execution, except with negligible probability.

We now consider the case that the receiver is the only party setting up timer(s). Again, we use the presumptive simulator $\mathcal{S}'$ to construct a legal environment $\mathcal{Z}'[\pi, \mathcal{D}]$ expecting to interact with the dummy adversary for which there is no simulator, contradicting the assumption. At the onset of the execution, the environment $\mathcal{Z}'[\pi, \mathcal{D}]$ samples $b \xleftarrow{\$} \{0,1\}$ and sends b as input to the honest committer. It also orders the adversary to corrupt the receiver and internally executes the simulator $\mathcal{S}$ for the corrupted *committer* to extract the commitment of the honest committer. Again, timers for the internally emulated simulator are handled relative to a presumptive real execution (independent of the honest committer's input b). In the real execution, the internally emulated simulator will succeed in extracting the correct bit with overwhelming probability. In the ideal execution, the extracted bit b' will either be independent of b or the extraction may fail altogether, allowing $\mathcal{Z}'$ to distinguish with non-negligible probability. □

In the next section, we will construct a TLUC-secure composable commitment scheme in the plain model where *both* parties use timers. In such a setting, a legal environment may not be able to internally execute the simulator without triggering a timeout for timers set up by honest parties.

3.5. Composable Commitment Schemes in the Plain Model

We are now ready to present our construction π_{MCOM} that TLUC-realizes the ideal functionality $\mathcal{F}_{\mathrm{MCOM}}$ (Figure 2.3) and prove its security. Our construction is based on the $\mathsf{UCC}_{\mathsf{OneTime}}$ commitment scheme in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model due to Canetti and Fischlin [CF01], which is a variant of the trapdoor commitment scheme due to Di Crescenzo, Ishai, and Ostrovsky [DIO98], which is in turn based on the commitment scheme due to Naor [N90].

In the original scheme $\mathsf{UCC}_{\mathsf{OneTime}}$, which is suitable for a single commitment only, the CRS consists of two parts: a pair of public keys (pk_0, pk_1) for a trapdoor PRG (Definition 3.5) as well as a uniformly random string $\sigma \in \{0,1\}^{4\kappa}$. With the knowledge of the associated PRG trapdoors, *i.e.* the secret keys (sk_0, sk_1), it is possible to extract the commitment. By changing the distribution of σ in an indistinguishable way, the commitment becomes equivocal.

To enable simulation in the case of static corruptions, the knowledge of only *one* trapdoor, depending on which party is corrupted, is sufficient. The other trapdoor does not even have to exist. Assuming trapdoor PRGs with dense public description (Definition 3.6), we can perform two coin-tosses to generate (pk_0, pk_1) resp. σ. While our coin-toss protocol (see Section 3.5.1) may, depending on the used commitment scheme, not be fully (straight-line) simulatable, it is straight-line simulatable if the simulator plays the initiator and the commitment scheme has a straight-line trapdoor property. This suffices to set up the extraction trapdoor if the sender in our composable commitment scheme is corrupted by having the commitment receiver, played by the simulator, start the coin-toss for (pk_0, pk_1). The simulator can equivocate the result to public keys for which it knows the secret keys. Conversely, the coin-toss for σ is started by the commitment sender. If it is honest, the simulator can simulate the coin-toss such that σ contains an equivocation trapdoor. From that point on, the original $\mathsf{UCC}_{\mathsf{OneTime}}$ scheme is executed, using the values obtained by this preamble phase instead of the CRS as in the original protocol. For each new commitment between two parties, the preamble phase is re-executed. A similar approach is used in [DMRV13].

Our coin-toss protocol π_{CT} uses the timed simulation-sound trapdoor commitment scheme SSCOM (see Section 3.3.2) whose equivocation trapdoor is protected by a timed commitment that can be extracted by the simulator. As SSCOM is timed simulation-sound, SSCOM commitments of corrupted committers remain binding in such a setting (if opened in time).

TLUC security does not imply concurrent self-composability. Thus, we cannot simply prove the security of a single commitment and conclude that it holds for multiple commitments performed concurrently. Indeed, when using weaker building blocks, our construction can be shown to securely realize one instance of $\mathcal{F}_{\mathrm{COM}}$, but not $\mathcal{F}_{\mathrm{MCOM}}$, where the latter captures concurrent self-composition.

In the following, we thus prove that π_{MCOM} TLUC-realizes the ideal functionality $\mathcal{F}_{\mathrm{MCOM}}$ for multiple commitments. We can thus plug π_{MCOM} into any (UC-secure) protocol making one subroutine call to $\mathcal{F}_{\mathrm{MCOM}}$, achieving TLUC security.

3.5.1. The Coin-Toss Protocol π_{CT}

One important building block towards constructing our TLUC-secure commitment scheme is the coin-toss protocol π_{CT} (Construction 2). It is essentially identical to the protocol due to Blum [B81], except for the use of a string commitment scheme and with the addition of handling the timers of SSCOM.

Construction 2 (Coin-Toss Protocol π_{CT})**.** Parameterized with a security parameter κ, a timed security parameter ℓ, a length parameter $s = s(\kappa)$ and a $\ell'(\ell(\kappa), \kappa)$-timed simulation-sound commitment scheme SSCOM with message space $M \supseteq \{0,1\}^s$.

1. On input $(\texttt{coin-toss}, \mathit{sid})$, the sender (also called *initiator*) samples $r \xleftarrow{\$} \{0,1\}^s$ uniformly at random.
2. Sender and receiver start an instance of SSCOM on common input $(1^\kappa, \texttt{commit}, \mathit{sid}, \ell'(\ell(\kappa), \kappa))$. The sender's private input for the commitment is r. All `notify` messages are forwarded between the adversary and the parties of SSCOM. Messages $(\texttt{timer}, \mathit{id})$ coming from a SSCOM party are forwarded to the adversary as $(\texttt{timer}, \mathit{id}, \ell)$, *i.e.* augmented with the timed security parameter ℓ.
3. After the commit phase has finished, the receiver samples $r' \xleftarrow{\$} \{0,1\}^s$ uniformly at random and sends (sid, r') to the sender.
4. Upon receiving (sid, r'), sender and receiver perform the unveil phase of SSCOM.
5. Upon successful completion, sender and receiver each output $(\texttt{coin-toss}, \mathit{sid}, r \oplus r')$ and halt otherwise.

As SSCOM may not be straight-line extractable, we cannot show that π_{CT} TLUC-realizes the coin-toss functionality $\mathcal{F}_{\mathrm{CT}}$. However, π_{CT} exhibits the following useful properties: If the commitment receiver is corrupted and the commitment scheme is equivocal, the coin-toss is simulatable. If the sender is corrupted and does not abort, the result of the coin-toss is distributed uniformly at random. Due to the simulation-soundness of SSCOM, the result of one session is independent of all other sessions of π_{CT} that may run concurrently, with the exception of aborts skewing the distribution.

We do not prove these properties on their own, but show them implicitly in the proof of the construction of the commitment scheme.

3.5.2. The Commitment Scheme π_{MCOM}

We now present the construction of the composable commitment scheme π_{MCOM}. We assume that the communication between protocol parties is ideally authenticated.

Construction 3 (Commitment Scheme π_{MCOM})**.** Parameterized by a timed security parameter $\ell(\kappa)$ and a trapdoor PRG PRG with key space $\{0,1\}^{l(\kappa)}$ for some polynomial l, domain $\{0,1\}^{\kappa}$ and range $\{0,1\}^{4\kappa}$.

Commit Phase.

1. Upon receiving $(\texttt{commit}, \mathit{sid}, \mathit{cid}, P_i, P_j, b)$ as input for the committer P_i, committer P_i and receiver P_j execute two instances of π_{CT} with timed security parameter $\ell(\kappa)$ to generate
   a) $(pk_0, pk_1) \in \{0,1\}^{l(\kappa)} \times \{0,1\}^{l(\kappa)}$ (the "extraction CRS") with the receiver acting as initiator in π_{CT} with session ID $(\mathit{sid}, \mathit{cid}, 0)$, where $l(\kappa)$ is the length of public keys of PRG.
   b) $\sigma \in \{0,1\}^{4\kappa}$ (the "equivocation CRS") with the committer P_i acting as initiator in π_{CT} with session ID $(\mathit{sid}, \mathit{cid}, 1)$.

   If both instances of π_{CT} terminate successfully, both parties store $(\mathit{sid}, \mathit{cid}, (pk_0, pk_1, \sigma))$. Otherwise, they halt the execution.
2. The committer P_i samples $r \xleftarrow{\$} \{0,1\}^{\kappa}$ and sets $c = \mathsf{PRG}(pk_0, r)$ if $b = 0$ and $c = \mathsf{PRG}(pk_1, r) \oplus \sigma$ if $b = 1$. Then, the committer sends $(\texttt{commitment}, \mathit{sid}, \mathit{cid}, c)$ to the receiver P_j. The committer stores $(\mathit{sid}, \mathit{cid}, (b, r, c))$, the receiver stores $(\mathit{sid}, \mathit{cid}, c)$ and outputs $(\texttt{committed}, \mathit{sid}, \mathit{cid}, P_i, P_j)$.

Unveil Phase.

1. Upon receiving $(\texttt{unveil}, \mathit{sid}, \mathit{cid}, P_i, P_j)$ as input, the committer P_i sends $(\texttt{unveil}, \mathit{sid}, \mathit{cid}, (b, r))$ to the receiver P_j.
2. Upon receiving $(\texttt{unveil}, \mathit{sid}, \mathit{cid}, (b, r))$ from the committer P_i, the receiver P_j checks if $c = \mathsf{PRG}(pk_0, r)$ for $b = 0$ or if $c = \mathsf{PRG}(pk_1, r) \oplus \sigma$ for $b = 1$, relative to the values stored for this sid and cid. If the check is successful, the receiver outputs $(\texttt{unveil}, \mathit{sid}, \mathit{cid}, P_i, P_j, b)$ and halts otherwise.

Theorem 3.5. *If* PRG *is a trapdoor PRG with dense public description and* SSCOM *is a (computationally) trapdoor, extractable and timed simulation-sound commitment scheme with appropriate parameters, then* π_{MCOM} *TLUC-realizes* $\mathcal{F}_{\mathrm{MCOM}}$ *in the presence of static corruptions.*

3.6. Proof of Security

In the following, we prove Theorem 3.5. For the sake of readability, we will often omit the parameters of random variables, ensembles or functions, in particular the security parameter κ or the (non-uniform) input z.

Proof. First, we define the simulator for π_{MCOM} and the dummy adversary.

Definition 3.10 (The Simulator $\mathcal{S}$)**.**

As we consider static corruptions, we distinguish between which parties are corrupted and which parties are honest in each session. For messages sent by (simulated) honest parties, we assume that the simulator reports these messages to the environment. Upon confirmation, the message is delivered. Conversely, the simulator delivers messages coming from the environment in the name of corrupted parties. For simulated honest parties waiting for a message, the internal simulation only continues when the message has been delivered.

Corrupted Receiver, Honest Committer. In case of a corrupted receiver and an honest committer, the simulator must be able to perform the commit phase without knowing the value v committed to. Later, upon learning v, it must be able to equivocate the commitment to v. To this end, it embeds an equivocation trapdoor.

1. Upon receiving the delayed output $(\texttt{committed}, \mathit{sid}, \mathit{cid}, P_i, P_j)$ from $\mathcal{F}_{\text{MCOM}}$, play the coin-toss for the extraction CRS honestly and report all messages. In case of an error or timeout, halt.

2. Sample $r_0, r_1 \xleftarrow{\$} \{0,1\}^\kappa$ and play the coin-toss for the equivocation CRS using the algorithm of the trapdoor committer C_{trap}. Equivocate the commitment such that the result of the coin-toss is $\sigma = \mathsf{PRG}(pk_0, r_0) \oplus \mathsf{PRG}(pk_1, r_1)$. In case of an error, halt.

3. Perform the commit phase by reporting $c = \mathsf{PRG}(pk_0, r_0)$ as message from the committer. Upon successful message delivery, allow the delayed output.

4. Upon receiving the delayed output $(\texttt{unveil}, \mathit{sid}, \mathit{cid}, P_i, P_j, b)$ from $\mathcal{F}_{\text{MCOM}}$, report $(\texttt{unveil}, \mathit{sid}, \mathit{cid}, (0, r_0))$ if $b = 0$ and $(\texttt{unveil}, \mathit{sid}, \mathit{cid}, (1, r_1))$ if $b = 1$. Upon successful message delivery, allow the delayed output.

Corrupted Committer, Honest Receiver. In case of a corrupted committer and an honest receiver, the simulator must be able to extract the value v committed to. To this end, it embeds an extraction trapdoor.

1. Generate keys $(pk_i, sk_i) \leftarrow \mathsf{PRG.TGen}(1^\kappa)$ for $i = 0, 1$ and perform π_{CT} for the extraction CRS using the algorithm of the trapdoor committer C_{trap}. Equivocate the commitment such that the coin-toss result is (pk_0, pk_1). In case of an error, halt.

2. Play the coin-toss for the equivocation CRS σ honestly and report all messages. In case of an error or timeout, halt.

3. Upon receiving $(\texttt{commitment}, \mathit{sid}, \mathit{cid}, c)$ from the environment:
   - Check, using sk_i, if c is in the range of $\mathsf{PRG}(pk_0, \cdot)$. If this is the case, send $(\texttt{commit}, \mathit{sid}, \mathit{cid}, P_i, P_j, 0)$ to $\mathcal{F}_{\text{MCOM}}$ on behalf of the corrupted committer P_i.
   - Otherwise, send $(\texttt{commit}, \mathit{sid}, \mathit{cid}, P_i, P_j, 1)$ to $\mathcal{F}_{\text{MCOM}}$ on behalf of the corrupted committer P_i.

   Subsequently, allow the public delayed output of $(\texttt{committed}, \mathit{sid}, \mathit{cid}, P_i, P_j)$ of the honest receiver.

4. Eventually, receive $(\texttt{unveil}, \mathit{sid}, \mathit{cid}, (b, r))$ from the environment. If c is a valid commitment to b and b agrees with the value sent to $\mathcal{F}_{\text{MCOM}}$, send $(\texttt{unveil}, \mathit{sid}, \mathit{cid}, P_i, P_j)$ to $\mathcal{F}_{\text{MCOM}}$ and allow the public delayed output. Otherwise, if $b = 1$ does not agree with the value sent to $\mathcal{F}_{\text{MCOM}}$ but $r \oplus \sigma$ is in the range of $\mathsf{PRG}(pk_1, \cdot)$, output a special error symbol $\bot_{\text{FAIL}}$. Otherwise, *i.e.* if the commitment is invalid, halt.

Both Parties Honest.
This case is identical to the case of the corrupted receiver, except that the simulator also plays the role of the honest receiver.

Both Parties Corrupted.
In this case, the simulator behaves like the dummy adversary.

We prove Theorem 3.5 by showing that the simulator given in Definition 3.10 is valid for the dummy adversary, which is sufficient (see Proposition 3.3). First, we define a series of hybrids for the proof, ordered by the start of the individual commitments. In order to better distinguish between elementary (non-UC) commitments in π_{CT} and commitments to be performed with π_{MCOM}, we also refer to the latter as *sessions*. Starting with an all-real execution, we consecutively introduce the simulator into the execution. Let $q = q(\kappa)$ denote an upper bound for the number of sessions.

For each hybrid, we show the indistinguishability of the environment's output compared to the previous hybrid. We also show that the environment is non-abusing, *i.e.* does not equivocate commitments, despite the equivocations performed by the simulator. While both properties are related, they require separate proofs at first. The hybrids are defined as follows:

- H_0: The real execution with the dummy adversary $\mathcal{D}$ and protocol π_{MCOM}.
- H_1^i: The execution with a simulator $\mathcal{S}_1^i$ and the protocol $\text{IDEAL}(\mathcal{F}_1)$ for the ideal functionality $\mathcal{F}_1$ that gives the inputs of all honest parties to the simulator and lets it determine all outputs. $\mathcal{S}_1^i$ honestly executes π_{MCOM} but uses the code of

the trapdoor committer C_{trap} of SSCOM whenever the simulator $\mathcal{S}$ would do so in the first i sessions. Outputs of simulated honest parties are forwarded to $\mathcal{F}_1$.

- H_2^i: Identical to H_1^q with $\mathcal{F}_2 = \mathcal{F}_1$ and $\mathcal{S}_2^i = \mathcal{S}_1^i$, apart from the following changes for sessions $k \leq i$:
  - If, in the k-th session, only the committer (of π_{MCOM}) is corrupted, $\mathcal{S}_2^i$ generates $(pk_j, sk_j) \leftarrow \mathsf{PRG.TGen}(1^\kappa)$ for $j \in \{0,1\}$ and equivocates the first coin-toss to (pk_0, pk_1).
  - If, in the k-th session, only the receiver (of π_{MCOM}) is corrupted or both committer and receiver are honest, $\mathcal{S}_2^i$ samples $r_0, r_1 \xleftarrow{\$} \{0,1\}^\kappa$, equivocates the second coin-toss to $\sigma = \mathsf{PRG}(pk_0, r_0) \oplus \mathsf{PRG}(pk_1, r_1)$ and commits by sending $c = \mathsf{PRG}(pk_0, r_0)$ as commitment.
- H_3^i: Identical to H_2^q, but $\mathcal{F}_3^i$ behaves like $\mathcal{F}_{\text{MCOM}}$ for the first i sessions and like $\mathcal{F}_2$ otherwise. $\mathcal{S}_3^i$ is identical to $\mathcal{S}_2^q$, except that it runs $\mathcal{S}$ for the first i sessions. If $\mathcal{S}$ outputs $\bot_{\text{FAIL}}$ in the j-th session, output $\bot_{\text{FAIL}}^j$.
- H_4: The ideal execution with the simulator $\mathcal{S}$ and $\mathcal{F}_{\text{MCOM}}$.

We now present the proof.

Claim 3.3. *If* SSCOM *is (computationally) trapdoor, then* out_1^i *and* out_1^{i+1} *are (computationally) indistinguishable. If* SSCOM *is timed simulation-sound, then for an appropriate (polynomial) timed security parameter* ℓ' *for* SSCOM*, the environment is non-abusing in* H_1^i*.*

Proof. We prove the indistinguishability by a reduction to the trapdoor property of SSCOM. Then, we show that the environment is non-abusing.

Indistinguishability. Let $\mathcal{A}'$ be the following adversary against the trapdoor property of SSCOM:

1. On input $(1^\kappa, z)$, $\mathcal{A}'$ internally starts an instance of H_1^i with input $(1^\kappa, z)$ for the environment.
2. If there is a commitment in the $i+1$-th session for which the trapdoor committer C_{trap} would be used, perform this commitment externally with the trapdoor experiment.
3. Eventually, $\mathcal{A}'$ outputs what the environment outputs.[13]

As the trapdoor property is not timed, $\mathcal{A}'$, running in polynomial time, constitutes a valid adversary. By definition of $\mathcal{A}'$, its advantage in the trapdoor game is identical to the distinguishing advantage of the environment between H_1^i and H_1^{i+1}, *i.e.*

[13]Without loss of generality, we may assume that the environment under consideration *always* outputs a bit.

$\mathrm{Adv}^{\mathrm{TD}}_{\mathcal{A}',\mathsf{SSCOM},\mathcal{O}}(\kappa, z) = |\Pr[\mathrm{out}^i_1 = 1] - \Pr[\mathrm{out}^{i+1}_1 = 1]|$, which is negligible as SSCOM is trapdoor. It follows that $|\Pr[\mathrm{out}^0_1 = 1] - \Pr[\mathrm{out}^q_1 = 1]| \leq q(\kappa) \cdot \mathsf{negl}^{\mathrm{TD}}_{\mathsf{SSCOM}}(\kappa)$ for a negligible function $\mathsf{negl}^{\mathrm{TD}}_{\mathsf{SSCOM}}$.

Non-Abusing. For the use in later hybrids, we need to show that the environment does not equivocate commitments where it is the committer, even if it receives equivocated commitments (*cf.* Definition 3.8). In analogy to [DMRV13], we call this property *non-abusing.* In order to show that the environment is non-abusing, we perform a reduction to the timed simulation-soundness of SSCOM. In the reduction, the reduction adversary $\mathcal{A}'$ internally executes an instance of the TLUC execution where the legal environment is abusing. We have to ensure that $\mathcal{A}'$ is able to internally simulate the execution without triggering a timeout in the reduction when there is no such timeout triggered by the TLUC environment. This is generally not trivial since the TLUC environment $\mathcal{Z}$ and the reduction adversary $\mathcal{A}'$ may count the number of performed computation steps differently. For example, $\mathcal{Z}$ may (legally) not count simulation overhead resulting from *e.g.* simulating other environments and protocols. Of course, $\mathcal{A}'$ has to count these steps. To address this caveat, $\mathcal{A}'$ runs, if applicable, the corresponding unrolled TLUC execution[14] where all steps are counted correctly, including the emulation overhead resulting from $\mathcal{A}'$ internally emulating said execution (*cf.* Proposition A.1).

Also, $\mathcal{Z}$ counts the steps relative to a (presumptive) execution of π_{MCOM} and the dummy adversary $\mathcal{D}$, while $\mathcal{A}'$ emulates a UC execution with an instance of $\mathrm{IDEAL}(\mathcal{F}_1)$ and a simulator $\mathcal{S}^i_1$. For the reduction to go through even in the presence of these discrepancies, the timed security parameter ℓ' of SSCOM has to be chosen sufficiently large to account for these differences.

Let $\mathrm{E}_{\mathrm{ABUSE}^i_1}$ denote the event that the environment is abusing in H^i_1, *i.e.* opens a SSCOM commitment to a value different from the extracted one.

Suppose for the sake of contradiction that $\Pr[\mathrm{E}_{\mathrm{ABUSE}^i_1}]$ is non-negligible, *i.e.* $\mathcal{Z}$ is abusing. Then, we can construct an adversary $\mathcal{A}'$ against the $\ell'(\ell(\kappa), \kappa)$-simulation-soundness of SSCOM using the following reduction:

1. On input $(1^\kappa, z)$, $\mathcal{A}'$ chooses $j \overset{\$}{\leftarrow} \{1, \ldots, q\}$ uniformly at random, where $q = q(\kappa)$ is an upper bound for the number of sessions. Then, $\mathcal{A}'$ internally executes an instance of H^i_1. `timer` and `notify` messages are forwarded between the internal execution of the UC experiment and the timed simulation-soundness game as needed. To this end, the timed security parameter ℓ is removed from `timer` messages coming from UC protocol parties and added to `timer` messages for UC parties.

[14]Informally, the unrolled execution of a routing environment is the execution where the routing environment is ignored and its internally emulated machines are executed directly. This removes the emulation overhead that would occur, while, due to the definition of routing environments, the output distribution remains identically distributed. For the definition of the unrolled execution, see Definition A.4

2. The commitments of the sessions that are played using C_{trap} in H_1^i are played as left sides with the timed simulation-soundness game, using the uniform distribution on bitstrings of length $l(\kappa)$ resp. 4κ.

3. If all or no parties in the j-th session are corrupted, abort.

4. If there is a corrupted party in the j-th session, play the single SSCOM commitment where the committer is corrupted as the right side in the simulation-soundness experiment.

Clearly, $\mathcal{Z}$'s view is identically distributed in the reduction and in H_1^i, unless all parties in the j-th session are honest or corrupted (and we do not need to argue the non-abusing property for this session). If the timed security parameter ℓ' is sufficiently large, $\mathcal{A}'$ never has to send a $(\texttt{notify}, \star, 1)$ message before $\mathcal{Z}$ (when asked through a `notify` message) does. Thus, any attack carried out by $\mathcal{Z}$ can be performed by $\mathcal{A}'$ without having to trigger a timeout prematurely. In particular, $\ell'(\ell(\kappa), \kappa)$ has to be sufficiently large to account for the following differences between the reduction and the presumptive execution according to which $\mathcal{Z}$ counts the number of steps performed:

- Overhead due to the reduction itself, *e.g.* interaction with the experiment and relaying of messages between the internal UC execution and the experiment.

- Overhead due to changes in the internally emulated UC execution, *e.g.* additional steps for IDEAL($\mathcal{F}_1$).

We want ℓ' to not depend on the internally emulated environment $\mathcal{Z}$ or the maximum number of sessions q, but only on the reduction as well as the timeout parameter ℓ[15]. To this end, we observe the following: $\mathcal{A}'$ relays messages between its internal UC execution and the experiment. For each commitment session, the overhead is independent of the number of sessions q.

$\mathcal{Z}$ may schedule the sessions in a way such that when the timer in the j-th session is active, multiple other sessions are activated. In the real execution, unless a timeout occurs, the number of sessions activated when the timer in the j-th has not expired is trivially upper-bounded by $\ell(\kappa)$, as each activation will consume at least one computation step. Thus, the maximum number of active sessions (for the timer in the j-th session) is independent of the bound for the number of challenge sessions $q(\kappa)$.

We have to consider the case that $\mathcal{Z}$ is a routing environment (*cf.* Appendix A.1.2). In this case, the steps needed by $\mathcal{A}'$ to emulate the TLUC execution may be more compared to the steps reported by $\mathcal{Z}$ due to unaccounted emulation overhead. However, according to Proposition A.1, there exists an "unrolled" execution (*cf.* Definition A.4) where there is no such emulation overhead and the environment's view is identically distributed. Due to the structure of routing environments, $\mathcal{A}'$ is able to "unroll" the execution in polynomial time before starting the internal emulation. At this point, no

[15]This does of course not rule out that ℓ may depend on $\mathcal{Z}$ and q. However, $\mathcal{Z}$ and q do not have to be *additionally* considered for ℓ'.

timers are active. Alternatively, we may provide $\mathcal{A}'$ with the necessary information via its advice.

Thus, setting ℓ' as $\ell(\kappa)$ plus the overhead for one single session plus some additional (small) overhead polynomial in κ, suffices. It follows that ℓ' is polynomially bounded, independent of q and the particular environment. As $\mathcal{A}'$ internally executes an instance of the TLUC execution with a legal environment and adversary and timeout parameter ℓ, $\mathcal{A}'$ is also a legal adversary against the $\ell'(\ell(\kappa), \kappa)$-simulation-soundness of SSCOM for appropriately chosen ℓ'. By definition of $\mathcal{A}'$, its advantage in the simulation-soundness experiment is equal to $\Pr[\mathrm{E}_{\mathrm{ABUSE}_1^i}]/q(\kappa)$ in H_1^i. Thus, a non-negligible probability of $\mathrm{E}_{\mathrm{ABUSE}_1^i}$ contradicts the simulation-soundness of SSCOM. It follows that $\Pr[\mathrm{E}_{\mathrm{ABUSE}_1^i}] \leq q(\kappa) \cdot \mathsf{negl}_{\mathsf{SSCOM}}^{\mathrm{SIMSOUND}}(\kappa)$ for a negligible function $\mathsf{negl}_{\mathsf{SSCOM}}^{\mathrm{SIMSOUND}}$ and that $\mathcal{Z}$ is non-abusing in H_1^i. □

Claim 3.4. *If* PRG *is a trapdoor PRG with dense public description and* SSCOM *is extractable, then* out_2^i *and* out_2^{i+1} *are computationally indistinguishable. Moreover, the environment is non-abusing in* H_2^i.

Proof. Let $H_2^{i,\#}$ be the sub-hybrid between H_2^i and H_2^{i+1} where only the first public key pk_0 is replaced with a public key originating from $\mathsf{PRG.TGen}$.

We start by proving the non-abusing property.

Non-Abusing. In order to show that the environment is non-abusing in H_2^i (the argument for $H_2^{i,\#}$ is analogous), we cannot directly reduce to the simulation-soundness of SSCOM as in the previous step. This is because we do not know the correct distribution of the equivocated commitment beforehand, as it depends on the second-round message of the coin-toss.

For $i = 0$, we have shown that the environment is non-abusing in $H_2^0 = H_1^q$. If the environment were to become abusing in H_2^1, we could use the environment to break the assumptions stated in Claim 3.4, namely the indistinguishability of keys output by $\mathsf{PRG.TGen}$ and uniformly random strings of appropriate length or the pseudorandomness of PRG, respectively. To this end, it is important that the reduction adversary is able to extract commitments created by the environment in order to determine if the environment is abusing in the first place. While SSCOM is not straight-line extractable, we require it to be extractable, *e.g.* via rewinding, in strict polynomial time. This can be achieved by using appropriate building blocks (*cf.* Corollary 3.1). As the properties we reduce to have non-interactive challenge phases, the reduction adversary is able to execute the extractor without having to rewind the reduction. (In other words, the extraction is trivially robust with respect to the "left sides" consisting of the experiments.) Thus, a strict polynomial-time reduction with only a polynomial loss of security is possible. As the aforementioned properties are not timed but hold for polynomial-time adversaries in general, we do not have to argue that the reduction adversary is able to provide its answer in time.

For $i > 1$, we can use this same strategy to create a contradiction to the assumptions of Claim 3.4 by using the fact that in the previous hybrid (*i.e.* H_2^i or $H_2^{i,\#}$), the environment was non-abusing. To this end, the adversary $\mathcal{A}'$ chooses a random index $j \xleftarrow{\$} [q]$ and extracts the environment's commitment in the j-th session. All messages in other sessions are answered as usual. If no party is corrupted or the extraction fails, the adversary outputs a random bit b. If the environment is abusing in the j-th session (determined by the same criterion as in Definition 3.8), it outputs 1, otherwise it outputs 0.

Thus, it holds that $\Pr[\mathrm{E}_{\mathrm{ABUSE}_2^i}] \leq \Pr[\mathrm{E}_{\mathrm{ABUSE}_2^0}] + 2 \cdot i(\kappa) \cdot q(\kappa) \cdot (2 \cdot \mathsf{negl}_{\mathsf{PRG}}^{\mathrm{PRGKEY}}(\kappa) + \mathsf{negl}_{\mathsf{PRG}}^{\mathrm{PRGVAL}}(\kappa) + 2 \cdot \mathsf{negl}_{\mathsf{SSCOM}}^{\mathrm{EXT}}(\kappa))$ for negligible functions $\mathsf{negl}_{\mathsf{PRG}}^{\mathrm{PRGKEY}}, \mathsf{negl}_{\mathsf{PRG}}^{\mathrm{PRGVAL}}$ and $\mathsf{negl}_{\mathsf{SSCOM}}^{\mathrm{EXT}}$ bounding an adversary's advantage distinguishing between uniformly random public keys and public keys originating from $\mathsf{PRG.TGen}$ resp. between random and pseudorandom values and the extraction error.

Note that in order to show the non-abusing property, it was not necessary to first show the indistinguishability of the adjacent hybrids. We will do this in the following, using the non-abusing property.

Indistinguishability, Corrupted Committer. Suppose for the sake of contradiction that out_2^i and $\mathrm{out}_2^{i,\#}$ are not computationally indistinguishable. We can then construct an adversary against the indistinguishability of uniformly random public keys[16] and public keys originating from $\mathsf{PRG.TGen}$.

The reduction adversary $\mathcal{A}'$ works as follows:

1. On input $(1^\kappa, z)$, internally start an execution of H_2^i with input $(1^\kappa, z)$.
2. Obtain the challenge pk_0 from the experiment and also sample $r \xleftarrow{\$} \{0,1\}^{l(\kappa)}$.
3. Equivocate the commitment in the i-th coin-toss such that it has the result (pk_0, r) and continue the execution of H_i^0 as specified.
4. Output what the environment outputs.

If pk_0 is a uniformly random string, then the environment's view is identically distributed as in H_2^i. If pk_0 originates from $\mathsf{PRG.TGen}$, then the environment's view is identically distributed as in $H_2^{i,\#}$. Thus, the distinguishing advantage of $\mathcal{A}'$ is the same as the environment's, contradicting the indistinguishability of PRG keys with trapdoor and uniformly random strings.

As the proof for the indistinguishability of $H_2^{i,\#}$ and H_2^{i+1} is similar, we omit it. All in all, as PRG keys with and without trapdoor are indistinguishable, we conclude that $|\Pr[\mathrm{out}_2^i = 1] - \Pr[\mathrm{out}_2^{i+1} = 1]| \leq 2 \cdot \mathsf{negl}_{\mathsf{PRG}}^{\mathrm{PRGKEY}}(\kappa)$ (where $\mathsf{negl}_{\mathsf{PRG}}^{\mathrm{PRGKEY}}$ is a negligible function bounding an adversary's distinguishing advantage between public keys originating from $\mathsf{PRG.TGen}$ resp. uniformly random strings of appropriate length), which is negligible.

[16]By Definition 3.6, we assume that uniformly random keys and keys output by $\mathsf{PRG.Gen}$ are computationally indistinguishable. Due to the transitivity of indistinguishability and the fact that both experiments have the same number of rounds, it suffices to consider uniformly random public keys in this reduction.

Indistinguishability, Corrupted Receiver. The proof of indistinguishability for a corrupted receiver is more involved, since reducing to the pseudorandomness property of the PRG leads to a change in the distribution of one PRG key. Let $H_2^{i,\text{PRGKEY}}$ be a sub-hybrid where only the PRG key is changed. As the proof for the indistinguishability of out_2^i and $\text{out}_2^{i,\text{PRGKEY}}$ is very similar as in the case of the non-abusing property (as we need to use the extractability property of SSCOM in order to embed the challenge), we omit it. Also, it is easy to see that the environment is non-abusing in this sub-hybrid. Let $H_2^{i,\text{PRGVAL}}$ denote the hybrid after $H_2^{i,\text{PRGKEY}}$ that is, with the exception of the PRG key distribution, identical to H_2^{i+1}.

Suppose for the sake of contradiction that $\text{out}_2^{i,\text{PRGVAL}}$ and $\text{out}_2^{i,\text{PRGKEY}}$ are not computationally indistinguishable, *i.e.* there exists a non-negligible function $\nu(\kappa)$ that lower-bounds $\mathcal{Z}$'s distinguishing advantage. We construct an adversary $\mathcal{A}'$ against the pseudorandomness property of PRG with non-negligible advantage $\nu'(\kappa)$.

$\mathcal{A}'$, on input $(1^\kappa, z)$, acts as follows:

1. Obtain the challenge public key pk from the PRG experiment as well as a challenge value s. Also, sample $pk' \xleftarrow{\$} \{0,1\}^{l(\kappa)}$.
2. Internally, start an instance of $H_2^{i,\text{PRGKEY}}$ with input $(1^\kappa, z)$. At the beginning of the i-th session, play the π_{CT} instance for the extraction trapdoor as follows: First, use the extractor E of SSCOM (*cf.* Corollary 3.1) to obtain the value v committed to by the environment. Simulate all protocol messages not related to SSCOM as in H_2^i. If E outputs $v = \bot$, *i.e.* the extraction fails, output a uniformly random bit to the PRG experiment. Otherwise, send $v \oplus (pk_0, pk_1)$ with $pk_b = pk'$ and $pk_{1-b} = pk$ as second-round message. Let v' denote the value unveiled by the environment. Let E_{INCON} denote the event that $v \neq v'$ or that the extraction fails. If E_{INCON} occurs, or $\mathcal{Z}$ does not open the commitment, output a uniformly random bit b'.
3. Sample $r \xleftarrow{\$} \{0,1\}^\kappa$ and equivocate the equivocation CRS σ to $\mathsf{PRG}(pk_b, r) \oplus s$.
4. Continue the execution, but commit as follows: If $b = 0$, send $\mathsf{PRG}(pk_0, r)$ as c. If $b = 1$, send $\mathsf{PRG}(pk_1, r) \oplus \sigma = s$ as c.
5. Finally, obtain the output b' of $\mathcal{Z}$ and output b'.

If s is a uniformly random string, then $\mathcal{Z}$'s view is distributed as in $H_2^{i,\text{PRGKEY}}$:

- σ is a uniformly random element (and so is $\sigma \oplus c$).
- If $b = 0$, c is always in the range of $\mathsf{PRG}(pk_0, \cdot)$.
- If $b = 1$, $c \oplus \sigma = s \oplus (\mathsf{PRG}(pk_1, r) \oplus s) = \mathsf{PRG}(pk_1, r)$ is always in the range of $\mathsf{PRG}(pk_1, \cdot)$.

If s is distributed pseudorandomly, *i.e.* is in the range of $\mathsf{PRG}(pk, \cdot)$, then $\mathcal{Z}$'s view is distributed as in $H_2^{i,\text{PRGVAL}}$:

- There exists $r' \in \{0,1\}^\kappa$ such that $\sigma = \mathsf{PRG}(pk_b, r) \oplus \mathsf{PRG}(pk, r')$
- If $b = 0$, c is always in the range of $\mathsf{PRG}(pk_0, \cdot)$.
- If $b = 1$, c is always in the range of $\mathsf{PRG}(pk_0, \cdot)$ and $c \oplus \sigma = s \oplus (\mathsf{PRG}(pk_1, r) \oplus s) = \mathsf{PRG}(pk_1, r)$ is always in the range of $\mathsf{PRG}(pk_1, \cdot)$.

If $\mathrm{E}_{\mathrm{INCON}}$ does not occur, $\mathcal{A}'$'s advantage in the PRG game is identical to $\mathcal{Z}$'s distinguishing advantage between $H_2^{i,\mathrm{PRGKEY}}$ and $H_2^{i,\mathrm{PRGVAL}}$. If $\mathrm{E}_{\mathrm{INCON}}$ occurs, $\mathcal{A}'$'s advantage is 0. We can thus use $\mathcal{A}'$ to bound the distinguishing advantage of $\mathcal{Z}$ as follows, using the pseudorandomness of PRG and the extractability of SSCOM:

$$\begin{aligned}
&|\Pr[\mathrm{out}_2^{i,\mathrm{PRGKEY}} = 1] - \Pr[\mathrm{out}_2^{i,\mathrm{PRGVAL}} = 1]| \\
\le\, &\Pr[\mathrm{E}_{\mathrm{INCON}}] + |\Pr[\mathrm{out}_2^{i,\mathrm{PRGKEY}} = 1 \wedge \neg\mathrm{E}_{\mathrm{INCON}}] - \Pr[\mathrm{out}_2^{i,\mathrm{PRGVAL}} = 1 \wedge \neg\mathrm{E}_{\mathrm{INCON}}]| \\
\le\, &\Pr[\mathrm{E}_{\mathrm{INCON}}] + |\Pr[\mathrm{out}_{\mathcal{A}'}^{s \text{ is uniformly random}} = 1 \wedge \neg\mathrm{E}_{\mathrm{INCON}}] - \\
&\Pr[\mathrm{out}_{\mathcal{A}'}^{s \text{ is pseudorandom}} = 1 \wedge \neg\mathrm{E}_{\mathrm{INCON}}]| \\
\le\, &\Pr[\mathrm{E}_{\mathrm{INCON}}] + \mathsf{negl}_{\mathsf{PRG}}^{\mathrm{PRGVAL}}(\kappa)
\end{aligned}$$

The last inequality holds due to the pseudorandomness of PRG.

As SSCOM is simulation-sound and extractable, we can bound $\Pr[\mathrm{E}_{\mathrm{INCON}}]$ by $\Pr[\mathrm{E}_{\mathrm{ABUSE}_2^{i,\mathrm{PRGKEY}}}] + \mathsf{negl}_{\mathsf{SSCOM}}^{\mathrm{EXT}}(\kappa)$ (as the execution is identically distributed to $H_2^{i,\mathrm{PRGKEY}}$ when the extraction occurs), where $\mathsf{negl}_{\mathsf{SSCOM}}^{\mathrm{EXT}}$ is a function bounding the extraction error.

Going from $H_2^{i,\mathrm{PRGVAL}}$ to H_2^{i+1} requires changing the PRG key distribution again and we omit the reduction.

It thus holds that

$$\begin{aligned}
&|\Pr[\mathrm{out}_2^{i} = 1] - \Pr[\mathrm{out}_2^{i+1} = 1]| \\
\le\, &\Pr[\mathrm{E}_{\mathrm{INCON}}] + \mathsf{negl}_{\mathsf{PRG}}^{\mathrm{PRGVAL}}(\kappa) + 2 \cdot \mathsf{negl}_{\mathsf{PRG}}^{\mathrm{PRGKEY}} \\
\le\, &\Pr[\mathrm{E}_{\mathrm{ABUSE}_2^0}] + 2 \cdot i(\kappa) \cdot q(\kappa) \cdot (2 \cdot \mathsf{negl}_{\mathsf{PRG}}^{\mathrm{PRGKEY}}(\kappa) + \mathsf{negl}_{\mathsf{PRG}}^{\mathrm{PRGVAL}}(\kappa) + 2 \cdot \mathsf{negl}_{\mathsf{SSCOM}}^{\mathrm{EXT}}(\kappa)) \\
&+ \mathsf{negl}_{\mathsf{SSCOM}}^{\mathrm{EXT}}(\kappa) + \mathsf{negl}_{\mathsf{PRG}}^{\mathrm{PRGVAL}}(\kappa) + 2 \cdot \mathsf{negl}_{\mathsf{PRG}}^{\mathrm{PRGKEY}} \\
\le\, &\Pr[\mathrm{E}_{\mathrm{ABUSE}_2^0}] + 2 \cdot q^2(\kappa) \cdot (4 \cdot \mathsf{negl}_{\mathsf{PRG}}^{\mathrm{PRGKEY}}(\kappa) + 2 \cdot \mathsf{negl}_{\mathsf{PRG}}^{\mathrm{PRGVAL}}(\kappa) + 3 \cdot \mathsf{negl}_{\mathsf{SSCOM}}^{\mathrm{EXT}}(\kappa))
\end{aligned}$$

which is negligible as the environment is non-abusing in H_2^0. Here, $\mathsf{negl}_{\mathsf{PRG}}^{\mathrm{PRGKEY}}$ denotes the function for the maximum advantage of an adversary distinguishing between uniformly random strings of appropriate lengths (*i.e.* PRG keys resulting from a coin-toss) as well as PRG keys that enable extraction resp. PRG keys originating from $\mathsf{PRG.Gen}$.

In particular, it follows that $|\Pr[\mathrm{out}_2^0 = 1] - \Pr[\mathrm{out}_2^q = 1]|$ is negligible too, as there exists a common bound for adjacent hybrids. □

Claim 3.5. *If the environment is non-abusing in H_2^q, then out_3^i and out_3^{i+1} are computationally indistinguishable. Moreover, the environment is non-abusing in H_3^i.*

Proof. As the only possible difference between H_3^i and H_3^{i+1} is the simulator outputting a special error symbol $\bot_{\mathrm{FAIL}}^{i+1}$ if the environment is able to open a commitment of a corrupted committer to a value different from the one extracted by the simulator in the $i+1$-th session, the proof is straight-forward. Let $\mathrm{E}_{\mathrm{FAIL}}^{i+1}$ denote this event.

Non-Abusing. If $\mathcal{Z}$ is non-abusing in H_2^q, then it is also non-abusing in H_3^{i+1}. This is due to the fact that out_3^i and out_3^{i+1} are identically distributed unless $\mathrm{E}_{\mathrm{FAIL}}^{i+1}$ occurs. If $\mathrm{E}_{\mathrm{FAIL}}^{i+1}$ occurs, the execution halts without giving $\mathcal{Z}$ the opportunity to equivocate a commitment that it has not been able to equivocate in the previous hybrid.

Indistinguishability. If $\mathrm{E}_{\mathrm{FAIL}}^{i+1}$ does not occur, H_3^i and H_3^{i+1} are identically distributed. It thus suffices to bound $\Pr[\mathrm{E}_{\mathrm{FAIL}}^{i+1}]$.

Corrupted Committer. If $\mathrm{E}_{\mathrm{FAIL}}^{i+1}$ occurs, then there exist $r_0, r_1 \in \{0,1\}^\kappa$ such that $\sigma = \mathsf{PRG}(pk_0, r_0) \oplus \mathsf{PRG}(pk_1, r_1)$ in the $i+1$-th session, which we call a collision. Let, for fixed pk_0 and pk_1, $S = \{\sigma' \mid \exists r_0, r_1 : \sigma' = \mathsf{PRG}(pk_0, r_0) \oplus \mathsf{PRG}(pk_1, r_1)\}$ denote the set of collisions and let $\mathrm{E}_{\mathrm{COL}}$ denote the event that $\sigma \in S$. As the size of PRG's image is upper-bounded by the size of its domain, *i.e.* 2^κ, we can bound $|S|$ by $2^{2\kappa}$. Thus, in a coin-toss for σ where the commitment is not equivocated and the second-round message is chosen uniformly at random, the probability $\Pr[\mathrm{E}_{\mathrm{COL}}] \leq 2^{2\kappa}/2^{4\kappa}$ is negligible.

In order for $\Pr[\mathrm{E}_{\mathrm{FAIL}}^{i+1}]$ to be non-negligible, the environment must bias the coin-toss either by only finishing sessions where the result is a collision or by equivocating its commitment such that a collision occurs. Let $\mathrm{E}_{\mathrm{INCON}}^{i+1}$ denote the event that the environment equivocates the commitment in π_{CT} in the $i+1$-th session. We can upper-bound bound $\Pr[\mathrm{E}_{\mathrm{FAIL}}^{i+1}]$ by $\Pr[\mathrm{E}_{\mathrm{COL}}^{i+1} \wedge \neg \mathrm{E}_{\mathrm{INCON}}^{i+1} \vee \mathrm{E}_{\mathrm{INCON}}^{i+1}]$. As $\mathcal{Z}$ is non-abusing in H_3^i, it follows that $\Pr[\mathrm{E}_{\mathrm{INCON}}^{i+1}] \leq \Pr[\mathrm{E}_{\mathrm{ABUSE}_2^q}]$, which is negligible. As $\Pr[\mathrm{E}_{\mathrm{COL}}] \leq 2^{-2\kappa}$, it follows that $\Pr[\mathrm{E}_{\mathrm{COL}} \wedge \neg \mathrm{E}_{\mathrm{INCON}}] \leq 2^{-2\kappa}$. All in all, we have $|\Pr[\mathrm{out}_3^i = 1] - \Pr[\mathrm{out}_3^{i+1} = 1]| = \Pr[\mathrm{E}_{\mathrm{FAIL}}^{i+1}] \leq 2^{-2\kappa} + \Pr[\mathrm{E}_{\mathrm{ABUSE}_2^q}]$, which is negligible as the environment is non-abusing.

As there exists a common bound for the distinguishing advantage between the hybrids H_3^i and H_3^{i+1} for every $i \in \{0, \ldots, q-1\}$, it follows that $|\Pr[\mathrm{out}_3^0 = 1] - \Pr[\mathrm{out}_3^q = 1]$ is negligible.

Corrupted Receiver. If the receiver is corrupted, the simulation is identical to the previous hybrid. Thus, $\mathcal{Z}$'s view is identically distributed and its distinguishing advantage is 0. $\square$

As H_4 is identical to H_3^q, except for syntactical changes subject to events with negligible probability (the final simulator outputs $\bot_{\mathrm{FAIL}}$ when the previous one would output $\bot_{\mathrm{FAIL}}^i$) it holds that out_4 and out_3^q are computationally indistinguishable.

As there exists a common bound for the distinguishing advantage between the individual hybrids, out_0 and out_4 are computationally indistinguishable and the claim follows. $\square$

Remark 3.6. Our proof differs from the one in [CF01] in a number of ways:

- In the proof for the $\mathsf{UCC}_{\mathsf{OneTime}}$ scheme, the reduction adversary has to guess the bit committed to by the honest committer when embedding the public key from the PRG reduction into the CRS. In our reduction, this is different as the reduction adversary knows the bit committed to by the honest party beforehand.
- The $\mathsf{UCC}_{\mathsf{OneTime}}$ scheme requires a PRG with a stretch of 3κ that expands strings of length κ to strings of length 4κ. This is due to the fact that the $\mathsf{UCC}_{\mathsf{OneTime}}$ simulator always embeds an equivocation trapdoor into the CRS, even if the receiver is honest. In our case, this is not only impossible if the committer is corrupted, but the analysis is also somewhat different, allowing the use PRGs with a stretch of 2κ only.

3.6.1. Generalizing Our Result

Our protocol π_{MCOM} uses a timed simulation-sound commitment scheme SSCOM to TLUC-realize $\mathcal{F}_{\mathrm{MCOM}}$. However, it can be generalized to realize $\mathcal{F}_{\mathrm{MCOM}}$ under other security notions, depending on the properties of SSCOM and the existence of certain cryptographic hardness assumptions.

If one desires a setting where there is no (even temporary) complexity asymmetry between simulator and environment, one can replace SSCOM with a UC-secure commitment scheme and obtains a commitment scheme that UC-realizes $\mathcal{F}_{\mathrm{MCOM}}$. Following the impossibility results of *e.g.* [CF01], this requires a setup.

The other extreme is to consider large complexity asymmetries that hold throughout the whole execution. This is the general setting of $(\mathcal{C}_{env}, \mathcal{C}_{sim})$-security introduced by [LPV09]. In this setting, $\mathcal{C}_{env}$ denotes the complexity class of the environment and $\mathcal{C}_{sim}$ the complexity class of the simulator. If $\mathcal{C}_{env} = \mathcal{C}_{sim}$, one achieves a framework with universal composition. In the case of UC security, the complexity class considered is (non-uniform) probabilistic polynomial time.

If $\mathcal{C}_{env}$ is *e.g.* (non-uniform) polynomial-time and $\mathcal{C}_{sim}$ is quasi-polynomial-time, then we can realize $\mathcal{F}_{\mathrm{MCOM}}$ in the plain model, assuming that trapdoor permutations with dense public description and subexponential hardness exist. Moreover, we require commitment schemes that are extractable in quasi-polynomial time.

Indistinguishability is even guaranteed if the environment passes its view after the execution to a distinguisher with complexity $\mathcal{C}_{sim}$, which is not usually guaranteed by other approaches.

In between these extremes, more fine-grained solutions are possible, *e.g.* by considering different security parameters for environment and simulator. In a very practical setting, one might believe that *e.g.* RSA-4096 is secure even for adversaries able to break RSA-1024. Our protocol can be adapted to this setting, using a simulator that is able to efficiently break RSA-1024 but not RSA-4096.

Remark 3.7. Our technique also weakens the assumptions for practical complexity leveraging: We can replace the timed commitment scheme with a “weak” commitment scheme that is initially hiding for every polynomial-time environment and adversary, but extractable for the simulator (that must not be able to break the other hardness assumptions used in the protocol). The security of this "weak" commitment scheme thus can be very low, as the simulation remains indistinguishable as long as the "weak" commitments remain hiding during their use in the coin-toss. Afterwards, they do not need to be hiding anymore.

3.6.2. Obtaining a Timer-Obeying Simulator Using a Setup

We have stated in Section 3.3.2 that a potential candidate for the weakly extractable timed commitment scheme TCOM used in SSCOM is the timed commitment scheme of Boneh and Naor [BN00], which is in the plain model.

In order to argue for the plausibility of their plain-model MPC protocol, Prabhakaran and Sahai [PS04] outline how to realize their Imaginary Angel in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model, essentially resulting in a UC protocol.

We discuss how to cast our protocol π_{MCOM} in the $\mathcal{F}_{\mathrm{KRK}}$-hybrid model (*i.e.* using a setup for *key registration with knowledge*) when using the timed commitment scheme of [BN00], resulting in a composable commitment that is simulatable even for simulators that obey to timers.

The changes to the commitment scheme of [BN00] are as follows: Instead of having the committer sample primes p_1, p_2 such that $p_1 \equiv p_2 \equiv 3 \mod 4$ and send $N = p_1 \cdot p_2$ to the receiver, we use the $\mathcal{F}_{\mathrm{KRK}}$ functionality to provide a public key N for each committer (to all parties) as well as the secret $\varphi(N)$ (to the committer only) according to the aforementioned distribution.

Informally, the extraction trapdoor $\varphi(N)$ is protected by the BBS assumption, which is believed to be hard for classic polynomial-time adversaries with only the knowledge of N. Otherwise, the protocol remains unmodified. Each N can even be used for multiple TCOM commitments [BN00], *i.e.* as sole setup for π_{MCOM}.

The new simulator does not have to use the costly (but still polynomial-time) `forced-open` algorithm, but can use $\varphi(N)$ to very efficiently extract timed commitments from malicious committers. Otherwise, the simulation remains unmodified. It is easy to see that the resulting simulator can obey all timers if the timed security parameter ℓ is chosen such that there is sufficient time for the timed commitment to be extracted using $\varphi(N)$ while still guaranteeing its timed hiding property without the knowledge of $\varphi(N)$.

While this modified protocol still crucially relies on timers to prevent the environment from extracting timed commitments created by the simulator and is thus not UC-secure, the simulation strategy is as "plausible" as any UC simulation strategy relying on a trusted setup.

3.7. Constant-Round Black-Box Composable General MPC

In order to achieve composable general MPC, we can plug the construction π_{MCOM} into any UC-secure general MPC protocol that makes one subroutine call to $\mathcal{F}_{\text{MCOM}}$[17] while maintaining security (using Corollary 3.5).

Hazay and Venkitasubramaniam [HV15] have presented a constant-round and black-box general MPC protocol in the $\mathcal{F}_{\text{CRS}}$-hybrid model based on public-key encryption and semi-honest oblivious transfer. Following the approach used in [BDH$^+$17], we can generate the CRS of the [HV15] protocol with a simulatable coin-toss, assuming that public-key encryption (PKE) schemes with indistinguishability under chosen plaintext attack (IND-CPA) security and oblivious public-key generation exist, thus casting the protocol in the $\mathcal{F}_{\text{MCOM}}$-hybrid model.

Theorem 3.6. *If timed commitment schemes with appropriate parameters and perfectly binding homomorphic commitment schemes as well as pseudorandom generators with dense public description and IND-CPA-secure PKE schemes with oblivious public-key generation exist, then for every well-formed*[18] *functionality $\mathcal{F}$, there exists a constant-round protocol $\pi_{\mathcal{F}}^{BB}$ in the plain model such that $\hat{\pi}_{\mathcal{F}}^{BB} \geq_{\text{TLUC}} \text{IDEAL}(\widehat{\mathcal{F}})$ and $\pi_{\mathcal{F}}^{BB}$ uses its building blocks in a black-box way only.*

In Theorem 3.6, $\hat{\mathcal{F}}$ denotes the multi-session existence of $\mathcal{F}$ (*cf.* [CR03]) that naturally captures concurrent self-composition.

Considering possible candidates for timed commitments and perfectly binding homomorphic commitment schemes, we obtain the following corollary.

Corollary 3.6. *If the generalized BBS assumption and the DDH assumption hold and trapdoor permutations with dense public description exist, then for every well-formed functionality $\mathcal{F}$, there exists a constant-round protocol $\pi_{\mathcal{F}}^{BB}$ in the plain model such that $\hat{\pi}_{\mathcal{F}}^{BB} \geq_{\text{TLUC}} \text{IDEAL}(\widehat{\mathcal{F}})$ and $\pi_{\mathcal{F}}^{BB}$ does not use non-black-box techniques.*

Alternative and more practically efficient constructions for composable general MPC in the plain model are possible. For example, one can use our composable commitment scheme to bootstrap the CRS of the OT protocol of Peikert, Vaikuntanathan, and Waters [PVW08]. The resulting protocol can then be combined with an arbitrary (efficient) UC-secure general MPC protocol in the $\mathcal{F}_{\text{OT}}$-hybrid model, resulting in a TLUC-secure general MPC protocol.

[17] As $\mathcal{F}_{\text{MCOM}}$ models multiple commitments between multiple parties, a single instance of $\mathcal{F}_{\text{MCOM}}$ is usually sufficient.

[18] Informally, a functionality $\mathcal{F}$ is *well-formed* if its behavior is independent of which parties are corrupted [CLOS02].

4. Updatable Composable Security

Parts of this chapter are based on an extended full version of the paper [BBK$^+$23] (© IACR 2023, DOI 10.1007/978-3-031-48624-1_19). In particular, [BBK$^+$23] and the full version do not include the notion of *Updatable UC Security* and the required technical enhancements.

Robin Berger has contributed a treatment of *timestamping*, which will be part of a future paper, but not part of this thesis. I would like to thank him for helpful discussions on the ideal functionality for updatable commitments. Brandon Broadnax has contributed to an earlier version of this work. I would also like to thank him for helpful discussions. Michael Klooß and I have equally contributed to game-based definitions and security notions concerning commitment schemes. While I have contributed through many technical discussions, the sections on rewinding as well as pseudo-oracles are overwhelmingly his contribution. In particular, it was his idea to use the commitment scheme of Goyal *et al.* [GLP$^+$15] instead of the construction of Canetti, Lin, and Pass [CLP10]. Discussions with Michael on other parts have been particularly helpful. Jörn Müller-Quade has contributed the overall idea, its technical aspects and, in particular, to the notion of Updatable UC security. Markus Raiber has contributed the general idea behind the security definition in Section 4.7.1 and provided preliminary versions of Definition 4.26 and Construction 6 that are not part of this thesis. I would also like to thank him for our discussions regarding Sections 4.7.2 and 4.7.3. Astrid Ottenhues has contributed by identifying possible instantiations. I would like to thank her for discussions on instantiations.

Abstract

Protocols for secure multi-party computation rely on assumptions such as an honest majority of protocol parties or the hardness of certain problems. These cryptographic hardness assumptions are usually believed to hold throughout and even after protocol execution. However, this is not necessarily true: A number of cryptographic assumptions, *e.g.* the security of SHA-1 or 512-bit RSA, turned out to be wrong in the first place or became insecure due to increased computational resources or algorithmic advances—even when the assumptions were still used. With the availability of universal quantum computers, this problem will be exacerbated.

Long-term security, introduced by Müller-Quade and Unruh (TCC 2007, JoC 2010), considers a setting where *all* hardness assumptions become invalid *after* the protocol execution has finished. While achieving such a strict notion is highly desirable, it is inadequate when assumptions may become invalid during protocol execution. Moreover, there is a class of building blocks, *e.g.* commitment schemes, whose security can be

preserved by "updating" to a new hardness assumption in time (*e.g.* from RSA-512 to RSA-4096) during protocol execution and whose security is not appropriately captured by long-term security.

To analyze the security of such protocols, we propose a variant of (long-term) UC security where hardness assumptions may adaptively become invalid at any point during or after a protocol execution. We call the resulting notion *Updatable Universal Composability.*

Unfortunately, the setting of updatable and composable security is subject to the known impossibilities of long-term security, ruling out the natural class of *long-term revealing* setup assumptions such as common reference strings to achieve updatable security. In order to circumvent this impossibility result, we make use of new techniques allowing rewinding-based simulation in a way that universal composability is possible. As a result, we are the *first* to construct a commitment scheme that is both statistically hiding and composable from only standard polynomial-time hardness assumptions in the CRS-hybrid model. Furthermore, we enhance the security of this commitment scheme with an *updatable* binding property. This is the first construction of an updatable composable commitment scheme.

Additionally, we construct protocols for composable zero-knowledge and commit-and-proof with long-term security. We also give an impossibility result for our setting, namely the impossibility of long-term-secure oblivious transfer. We also construct protocols for composable reactive two-party computation with long-term security for one party, which is the best one can hope for in our setting.

4.1. 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.

Unless an honest majority [BGW88] or very strong setup assumptions such as trusted initializers [B97] or tamper-proof hardware tokens [DKM11] exist, building blocks such as one-way functions or semi-honest oblivious transfer are necessary for MPC [N91; K88]. Unfortunately, it is not known if hard problems yielding these building blocks actually exist—if this question could be answered one way or the other, the consequences for complexity theory would be huge. In the meantime, the best we can do is to resort to making assumptions about the hardness of certain problems. Often, the hardness is based on or related to the perceived (but unproven) hardness of mathematical problems, *e.g.* the RSA problem. In other cases, the arguments are more of a heuristic nature, *e.g.* in the case of hash functions or block ciphers. Historically, a number of widely-used candidates for cryptographic building blocks has either turned out to be insecure in the first place (*e.g.* MD5 [K06] or SHA-1 [SKP16]) or to be inadequate because a) the concrete choice of the "security parameter" has become too low due to advancements in computational performance (*e.g.* in the case of DES with 56 bit key size) or b) algorithmic advancements have led to new assessments of the security of certain parameter sets (*e.g.*

the security estimates of RSA [RSA78]). In the face of the possible advent of (universal) quantum computers, the danger of assumptions and thus protocols becoming insecure will be exacerbated. Indeed, many of today's hardness assumptions do not hold for bounded-error quantum polynomial-time (BQP) algorithms, and further advances are hard to predict.

If an assumption turns out to be invalid, one is faced with a number of problems. First of all, an adversary who is in possession of protocol messages may obtain past secrets. Given the actions of intelligence agencies that are known to capture large amounts of traffic and store it for later processing, this is a real threat, even if the parties participating in a protocol are honest.

This setting where (all assumed) cryptographic hardness assumptions hold during protocol execution but *all* hardness assumptions become invalid is captured by the notion of *long-term* or *everlasting* security [MU10]. While long-term security is often possible for a subset of protocol parties, *e.g.* for commitment schemes that statistically protect either the committer or the receiver, giving *all* parties this strong level of security may be impossible.

However, we notice that some security properties like the soundness of a proof system or the binding property of a commitment scheme are only important while the protocol is still executed. For such properties, computational security that can be "updated" to new hardness assumptions may be sufficient for many applications. Taking a closer look, it turns out that merely requiring "cryptographic agility" [ABBC10; OP19], *i.e.* being able to use different cryptographic algorithms for a specific task, is often insufficient: When updating *e.g.* the binding property of a commitment scheme, it is crucial that the update is performed *consistently*, preventing a malicious committer from committing to a different value during the update. This example illustrates that the update process not only must already be taken care of at protocol design time, but also needs to be considered when defining and analyzing a protocol's security.

Updating cryptographic protocols, *e.g.* commitment schemes [BGB17] or capturing security in a setting where some cryptographic hardness assumption become invalid [BFG19] have been considered before. However, previous approaches only consider a stand-alone setting where only one protocol is executed at any time. In a setting where multiple, possibly different protocols may be executed concurrently, stand-alone security is insufficient. It is well known that there exist protocols that are secure when executed on their own, but become completely insecure when two instances of the same protocol are executed in parallel [GK90].

While the notion of long-term security [MU10] implies universal composability, it is too weak in the sense that it cannot capture cryptographic hardness assumptions becoming invalid during protocol execution, and, at the same time, too strong in the sense that *all* hardness assumptions eventually become invalid. Thus, it is not suitable for the analysis of protocols with updatable security. This leads to our first research question:

How can we generalize the notion of long-term security to a setting that facilitates the analysis of protocols with updatable security, in particular where a) some *hardness assumptions may become invalid already during protocol execution, while b) other hardness assumptions may continue to be valid after the protocol execution has finished?*

The notion of long-term security faces a number of impossibility results. In particular, many commonly used and natural setup assumptions (*e.g.* common reference strings) are too weak to construct long-term-secure composable commitment schemes, *i.e.* commitment schemes that are both long-term hiding and computationally binding as well as straight-line equivocal and extractable.

As we will see, composable and updatable commitment schemes need to feature very similar properties, and indeed, the impossibility results of long-term security apply to our setting, too, raising 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 long-term-secure composable commitment schemes in the CRS-hybrid model. In order to circumvent this impossibility result, we propose new techniques and a new security notion based on UC security which covers the possibility of extracting statistically hiding commitment schemes via rewinding. While it may sound counterintuitive at first, our notion remains closed under protocol composition like UC security.

Using this long-term-secure composable commitment scheme, we construct the *first* composable commitment scheme whose binding property can be *updated* to new hardness assumptions. Based on these updatable and long-term-secure commitment schemes, we construct a variety of protocols achieving long-term or updatable security.

4.1.1. Outline and Contribution

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

1. 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 4.3. As the committed value is only computationally determined in this setting, we use a rewinding-based committed-value "oracle" $\mathcal{O}_{\mathsf{CCA}}$. To enable a formal analysis in this setting, we introduce *pseudo-oracles* and show important properties of these.

2. We extend the notions of Universal Composability [C01] and long-term security [MU10] to a setting where cryptographic hardness assumptions may become invalid at any point in the execution in Section 4.4. Like in [CLP10], we enable the extraction of commitments through a helper. In contrast to [CLP10], the helper carefully uses rewinding for extraction. As the helper is accessible by environment

and adversary, we retain universal composability. We call the resulting security notion *Updatable Universal Composability.*

We provide several justifications for our notion in Section 4.4, *i.e.* can show that our notion implies established security notions for large classes of protocols. Examples include UC security or stand-alone real-ideal security, *i.e.* notions that do not feature the helper $\mathcal{H}$. Intuitively, this shows that the helper does not affect the security of these protocol classes. Moreover, our notion features environmental friendliness [CLP13a] for a large class of security properties, similar to the notion of [CLP10].

3. In Updatable UC, we construct a composable commitment scheme which is *long-term-secure* in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model, circumventing the impossibility results of Müller-Quade and Unruh [MU10] in Section 4.5. To the best of our knowledge, we are the *first* to achieve this strong notion of security for commitment schemes without resorting to hardware assumptions (or assumptions that are sufficient for statistical UC security).

4. In Section 4.5, we also extend the long-term-secure commitment scheme to an *updatable* commitment scheme that repeatedly allows to update the computational binding property to new hardness assumptions.

5. We present several applications of the commitment schemes in our framework. We obtain composable oblivious transfer with long-term security for one party in Section 4.7. As we will show in Section 4.4, this is provably the best possible security in our setting. We extend the protocol for oblivious transfer to a protocol for composable reactive general two-party computation with long-term security for one party.

   Further applications include zero-knowledge or commit-and-prove, which we again realize with long-term security for both parties.

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

4.1.2. Related Work

Long-Term Security. As outlined in the introduction, the notion of long-term security as introduced by Müller-Quade and Unruh [MU10] is closely related to our setting. In [MU10], a composable long-term-secure commitment scheme is constructed from a trusted signature card. Conversely, large classes of setup assumptions, namely *long-term revealing* setups (Definition 2.11), have been shown to not admit long-term-secure composable commitment schemes. In contrast to standard UC security [CLOS02], where composable general MPC protocols can be constructed in the $\mathcal{F}_{\mathrm{COM}}$-hybrid model assuming semi-honest stand-alone-secure oblivious transfer, long-term-secure composable commitment schemes do not admit composable general multi-party computation with long-term security.

Composable long-term-secure commitment schemes also have previously been constructed from other hardware assumptions, *e.g.* fully malicious physical unclonable functions (PUFs) (together with a CRS) [MMSU22].

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 [B97]. However, if the party providing the randomness is untrusted or becomes corrupted at any point, all security may be 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, at the expense of composability (see Remark 4.33 for a discussion). The same holds for many composable commitment schemes with statistical security for one party, *e.g.* [DN02], as discussed and proven in [MU10].

Breakdown Resilience. Brendel, Fischlin, and Günther [BFG19] have extended the established stand-alone security model for key exchange [BR94] to a setting where the adversary may adaptively break the security of primitives or assumptions used in the protocol under analysis. While our motivation for the setting considered is similar, the technical details differ: In [BFG19], the proposed *break oracle* for the adversary is highly stateful and closely linked with the protocol execution. For example, the oracle may allow the adversary to program an idealized hash function or return the secret keys to all previously generated public keys. The same holds for explicit hardness assumptions such as the discrete logarithm (DLOG) assumption, where the oracle of [BFG19] only provides discrete logarithms of values previously generated through an explicit $\mathsf{GroupExp}$ operation. In contrast, we envision our "complexity oracles" to be deterministic and stateless and, in particular, independent of the current execution's state. For an example oracle in our framework, see Definition 4.6.

Reconfigurable Cryptography. Proposed by Hesse, Hofheinz, and Rupp [HHR16], *reconfigurable cryptography* allows the update of encryption and signature schemes by changing a single central parameter in a public key infrastructure. The authors propose concrete schemes and hardness assumptions in a stand-alone setting. In contrast to our goal, Hesse, Hofheinz, and Rupp [HHR16] aim to protect the security of long-term secrets that are used to derive short-term keys based on "reconfigurable" assumptions. Protecting the security of ciphertexts or signatures created with short-term keys is not intended.

Updatable Commitment Schemes. First introduced as *commitment schemes with prolongable binding* by Demirel and Lancrenon [DL15], the authors of [DL15] intend to prolong the security of statistically hiding commitment schemes by recommitting to the same value in a consistent way using a new and stronger commitment scheme. In [DL15], this update is performed for the Pedersen commitment scheme [P92] using a higher security parameter while proving consistency using a novel perfect zero-knowledge proof scheme.

Cabarcas *et al.* [CDG$^+$15] construct updatable commitment schemes that are post-quantum-secure. Again, consistency is achieved using a suitable proof scheme. Their construction updates a specific DLOG-based (Pedersen) commitment to a specific (plausibly post-quantum-secure) lattice-based commitment.

Buldas, Geihs, and Buchmann [BGB17] propose a different approach to achieve consistency, namely by repeatedly committing to the message and to the decommitments of all previous commitments. We use the same technique for our updatable commitment scheme in Section 4.5. They prove the security of their construction in the concrete security setting and also introduce the notion of *extractable-binding* security for commitment schemes, which we adapt to our setting in Definition 4.14.

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; GLP$^+$15] or zero-knowledge [K20; OOR$^+$14]. To this end, a very helpful tool is a robust extraction lemma [GLP$^+$15], which allows rewinding-based extraction without disturbing “left sides” up to a certain round complexity. We also use a variant of [GLP$^+$15] (*cf.* Appendix A.2.2).

Allowing a UC simulator to rewind the execution has first been proposed by Nielsen [N03], leading to a security notion with properties reminiscent of SPS security, namely limited composability.

Canetti, Lin, and Pass [CLP10] have proposed the notion of chosen commitment attack (CCA) security for commitment schemes and also given the first construction. Informally, CCA security guarantees the hiding property even for adversaries that have access to a committed-value oracle that extracts commitments created by the adversary, subject to the condition that the *identity* or *tag* of the challenge commitment is never used when querying the oracle. The commitment scheme in [CLP10] can be extracted either straight-line by inefficient computations or efficiently using rewinding. With 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*, that allows corrupted parties to extract their own commitments. By also giving the environment access to the helper, the notion achieves universal composability.

A drawback of the proposed approach is that UC compatibility is limited. Informally, 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 is also secure under their notion, at the expense of a higher round complexity of the CCA-secure commitment scheme. This has subsequently been improved, *e.g.* by Canetti, Lin, and Pass [CLP13a].

We use an approach that is reminiscent of the techniques of [CLP10]. In particular, we also use a helper to enable 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 adjusted definitions as well as changes to the execution experiment. For details, see Sections 4.3 and 4.4.

4.2. Definitions

We now discuss preliminaries relevant to this chapter and provide necessary definitions.

4.2.1. Some (Machine) Modelling Details

Composition and interaction of machines can be specified very abstractly [MR11] or very concretely [C01]. In Sections 4.2 and 4.3, 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 machines 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 $\mathcal{A}$.

Definition 4.1 (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 every unbounded distinguisher $\mathcal{D}$ which gets black-box (rewinding) access to $\mathcal{A}_0$ or $\mathcal{A}_1$, there exists a negligible function negl such that for every security parameter $\kappa \in \mathbb{N}$ and every $z \in \{0,1\}^*$, the distinguishing advantage of $\mathcal{D}$ on input $(1^\kappa, z)$ is 0 (resp. bounded by $\mathsf{negl}(\kappa)$). 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 4.1. *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 $\mathcal{O}$.*

4.2.2. Typed Commitment Schemes

In our constructions, we often need to consider *joint properties* of commitment schemes $\mathsf{COM}_1, \dots, \mathsf{COM}_n$. To formalize this setting (and simplify its notation), we introduce *typed commitment schemes*. These are minor variations of ordinary commitment schemes parameterized with a *type parameter* $\mathcal{T}$, denoting which type is to be used. Ordinary commitment schemes are recovered by choosing $\mathcal{T} = \{0\}$. Unless stated otherwise, we consider commitment schemes with setup.

In contrast to the definitions in Section 2.3.2, we employ a slightly different syntax in the following.

Definition 4.2 (Typed Commitment Schemes)**.** A *typed commitment scheme (with setup)*, non-interactive unveil phase, message space M and type space $\mathcal{T}$ is a tuple $(\mathsf{Setup}, \mathsf{C}, \mathsf{R})$, where

1. Setup is a PPT algorithm which on input $(1^\kappa, t)$ outputs a commitment key ck for type $t \in \mathcal{T}$.
2. $\langle \mathsf{C}, \mathsf{R} \rangle$ is an interactive protocol with PPT machines.
3. The protocol has two phases: a *commit phase* and an *unveil phase*. In both phases, C and R receive common input $(1^\kappa, \mathrm{ck}, t)$, where κ is a security parameter, $t \in \mathcal{T}$ is the type, and ck the commitment key (for type t). C additionally receives a private input $v \in M$ to be committed.
4. The commit phase results in a joint output c, called the commitment, a private output d for C, called *decommitment* or *unveil information*. Without loss of generality, c can be the full transcript of the interaction between C and R. As a shorthand to refer to the type of c, we write $\mathsf{type}(c)$.
5. In the unveil phase, committer C sends the pair (v, d) to the receiver R, which decides to accept or reject the decommitment or opening (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 $\mathsf{OPEN}(\mathrm{ck}, c, v, d) = 1$, and rejects otherwise.

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

Definition 4.3 (Stateless and Public-Coin Receivers)**.** A commitment scheme $\langle \mathsf{C}, \mathsf{R} \rangle$ 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, additionally, 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.

The definitions of the binding and hiding property are very similar to the ones in Section 2.3.2 and therefore omitted.

4.2.3. Cryptographic Hardness Assumptions

We want to analyze security in a setting where cryptographic hardness assumptions may become invalid throughout a protocol execution. While the exact definition of what is considered to be a cryptographic hardness assumption is not important, we re-state definitions from [GK16] as possible examples.

Definition 4.4 (Privately-Verifiable Search Complexity Assumption, adapted from [GK16])**.** An assumption is a *privately-verifiable search complexity assumption* if it consists of a pair of probabilistic polynomial-time algorithms $(\mathcal{D}, \mathcal{R})$, and it asserts that for any efficient algorithm $\mathcal{M}$ there exists a negligible function negl such that for every $\kappa \in \mathbb{N}$ and $z \in \{0,1\}^*$,

$$\Pr[\mathcal{R}(x, y, r) = 1 \mid x \leftarrow \mathcal{D}(1^\kappa; r); y \leftarrow \mathcal{M}(1^\kappa, z, x)] \le \mathsf{negl}(\kappa)$$

where the probability is over the coins of $\mathcal{D}$ and $\mathcal{M}$.

Examples of search complexity assumptions include factoring, the RSA assumption and its variants, the DLOG assumption or the LWE assumption [GK16].

To capture decision-based assumptions such as the decisional Diffie-Hellman assumption or the quadratic residuosity (QR) assumption, the following definition is useful [GK16]:

Definition 4.5 (Decisional Complexity Assumption, adapted from [GK16])**.** An assumption is a *decisional complexity assumption* if it is associated with two probabilistic polynomial-time distributions $(\mathcal{D}_0, \mathcal{D}_1)$, such that for any efficient algorithm $\mathcal{M}$ there exists a negligible function negl such that for every $\kappa \in \mathbb{N}$ and $z \in \{0,1\}^*$,

$$\Pr[\mathcal{M}(1^\kappa, z, x) = b \mid b \overset{\$}{\leftarrow} \{0,1\}, x \leftarrow \mathcal{D}_b(1^\kappa)] \le \frac{1}{2} + \mathsf{negl}(\kappa)$$

where the probability is over the choice of b, the coins of $\mathcal{D}_b$ and the coins of $\mathcal{M}$.

In order to model that a hardness assumption becomes invalid, we provide the adversary with a (deterministic and stateless) "complexity oracle" (Section 4.3.1) for the problem. It is parameterized with the security parameter, receives auxiliary input z and performs computations on the input provided by the adversary. For example, we consider the following oracle for the RSA problem:

Definition 4.6 (RSA Complexity Oracle)**.** Let Gen be a parameter generation algorithm for the RSA problem that, on input 1^κ, generates instances of the RSA problem with security parameter κ. $\mathcal{O}_{\mathrm{RSA}}$ is defined as follows:

1. On input $(1^\kappa, z, N, e, J)$, check that $(N, e) \in \mathrm{Im}(\mathsf{Gen}(1^\kappa)) \wedge \gcd(J, N) = 1$ and abort otherwise.
2. Output x such that $x^e \equiv J \mod N$. If no such x exists, output a special error symbol $\perp$.

Discussion. Throughout this chapter, the adversary will be provided with various complexity oracles. Before the adversary has access to an oracle for a problem $\mathcal{P}$, we usually assume that $\mathcal{P}$ is hard for PPT adversaries without access to complexity oracles. This is in line with how hardness assumptions are usually used in cryptographic protocols. The case of relative hardness is analogous.

While the formalism can deal with probabilistic stateful oracles, we only consider deterministic stateless complexity oracles in order to avoid problems when an adversary with access to a complexity oracle is rewound. For a discussion, see Section 4.3.2.

On the one hand, oracles may not only solve specific problems, but may encompass whole complexity classes (*e.g. stateless* BQP with classical input and output, EXP or PSPACE). On the other hand, they may be limited to very specific tasks, *e.g.* the finding of hash collisions.

We are also interested in *relative hardness*. Possible examples include the RSA assumption and post-quantum assumptions such as LWE [R05]: If LWE is hard for BQP adversaries, then LWE is also hard for adversaries with access to an RSA oracle, as the RSA problem is in BQP [S94]. Conversely, there exist assumptions which are not believed to be relatively hard, *e.g.* DLOG in $\mathbb{Z}_p$ and the RSA assumption.

Remark 4.1 (Quantumness)**.** Due to our reliance on *rewinding* of the adversary, we cannot handle stateful (quantum) oracles well. However, our techniques allow to handle stateless (quantum) oracles with classical input and output, *e.g.* factoring or DLOG oracles from [S94] as well as *e.g.* an oracle for Grover's algorithm [G96].

4.3. Concurrently Extractable Trapdoor Commitment Schemes

In this section, we define security notions of commitment schemes, in particular, security in the presence of a "complexity oracle", which will be denoted as *enhanced* security. We begin by outlining the notion of complexity oracles. Then we define pseudo-oracles, which capture oracles which use rewinding. With these preliminaries at hand, we define enhanced CCA security of commitment schemes. Lastly, we construct a commitment scheme which is computationally CCA-binding and CCA-trapdoor (and, in particular, statistically CCA-hiding).

4.3.1. Complexity Oracles

To model hardness assumptions becoming invalid, we augment PPT adversaries with a *complexity oracle* that can be used to solve problems that (are believed or assumed to be) hard for (non-uniform) PPT adversaries. For example, a complexity oracle could solve instances of the RSA problem or execute (stateless) BQP algorithms with classical input or output.

The complexity oracles we consider consist of a "shell" complexity oracle $\mathcal{O}_{\mathsf{comp}}(1^\kappa, z)$, which encapsulates a set of (specific) *stateless* complexity oracles $\mathcal{O}_{\text{name}}$. For this, $\mathcal{O}_{\mathsf{comp}}$ responds to $(\texttt{oracle}, \text{name}, m)$ with $\mathcal{O}_{\text{name}}(1^\kappa, z, m)$ (if $\mathcal{O}_{\text{name}}$ exists, else $\perp$), where κ

is the security parameter and z is some (possibly non-uniform) auxiliary input. We have two interpretations of $\mathcal{O}_{\mathsf{comp}}$.

The *stateless* interpretation separates the stateful bookkeeping of $\mathcal{L}$ into another party $\mathcal{H}_{\mathcal{L}}$ as follows:

- Party $\mathcal{H}_{\mathcal{L}}$: Upon receiving $(\texttt{invalidate}, \text{name})$, add name to $\mathcal{L}$, thereby invalidating assumption name. Initially, $\mathcal{L}$ is empty.
- Oracle $\mathcal{O}_{\mathsf{comp}}(1^\kappa, z)$: Upon receiving $(\texttt{oracle}, \text{name}, m)$, respond with $\mathcal{O}_{\text{name}}(1^\kappa, z, m)$ (if $\mathcal{O}_{\text{name}}$ exists, else $\perp$).

Since $\mathcal{O}_{\mathsf{comp}}$ is stateless (and cannot check if name $\in \mathcal{L}$), we have to restrict to admissible adversaries (or algorithms) for $\mathcal{O}_{\mathsf{comp}}$ in this setting.

Definition 4.7. Let $\mathcal{O}_{\mathsf{comp}}$ be a stateless complexity oracle with bookkeeping party $\mathcal{H}_{\mathcal{L}}$. An *admissible* adversary $\mathcal{A}$ for $\mathcal{O}_{\mathsf{comp}}$ only queries the $\mathcal{O}_{\mathsf{comp}}$ with $(\texttt{oracle}, \text{name}, m)$ if name $\in \mathcal{L}$, *i.e.* if name was previously invalidated.

In the *stateful* interpretation of $\mathcal{O}_{\mathsf{comp}}$, $\mathcal{O}_{\mathsf{comp}}$ implements $\mathcal{H}_{\mathcal{L}}$ and enforces the restrictions, *i.e.* $\mathcal{O}_{\mathsf{comp}}(1^\kappa, z)$ handles two types of messages:

- Upon receiving $(\texttt{invalidate}, \text{name})$, add name to $\mathcal{L}$, thereby invalidating assumption name. Initially, $\mathcal{L}$ is empty.
- Upon receiving $(\texttt{oracle}, \text{name}, m)$, check if name $\in \mathcal{L}$. If true, then respond with $\mathcal{O}_{\text{name}}(1^\kappa, z, m)$ (if $\mathcal{O}_{\text{name}}$ exists, else $\perp$), else return $\perp$.

Clearly, any adversary interacting with stateful $\mathcal{O}_{\mathsf{comp}}$ can be transformed into an admissible adversary interacting with stateless $\mathcal{O}_{\mathsf{comp}}$ and $\mathcal{H}_{\mathcal{L}}$.

Remark 4.2. Usually, complexity oracles are supposed to solve problems related only to the current security parameter (*e.g.* RSA instances of size κ (but not larger instances, *e.g.* of size κ^2), or exponential-time computations bounded by 2^κ). Thus, $\mathcal{O}_{\mathsf{comp}}$ receives the security parameter as first unary input. Moreover, $\mathcal{O}_{\mathsf{comp}}$ receives non-uniform input like other adversarial parties.

Remark 4.3. We consider only *deterministic stateless* oracles, because they clearly behave well under rewinding. See Appendix A.2.3 for a discussion and how to relax the restriction to probabilistic stateless oracles.

4.3.2. 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 core 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, appropriately designed 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 4.5 for a brief discussion. In particular, all our constructions should be robust to changes to the definition of pseudo-oracles.

Definition 4.8 ((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 4.4 (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.

Remark 4.5. Our definition of pseudo-oracles 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 a rewinding-based setting.

A naïve but natural alternative definition is the following: A pseudo-oracle $\mathcal{O}$ might 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 makes any reasoning about "$\mathcal{A}^{\mathcal{O}}$" basically impossible. For example, $\mathcal{A}^{\mathcal{O}}$ may behave entirely differently depending on $\mathcal{O}$. Hence, for such a relaxed notions, oracle-like behavior must explicitly be imposed, *e.g.* by requiring that the behavior of "$\mathcal{A}^{\mathcal{O}}$" is explainable by $\mathcal{A}$ receiving oracle responses. While such conceptual relaxations may be of interest, they seem harder to tame than our more restrictive approach, and unless applications are known which require them, they offer no real advantage.

In summary, we intentionally choose a notion of pseudo-oracles 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).

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 (pseudo-)oracle should have.[1] In our setting, all pseudo-oracles of interest are *black-box*.

[1] For example: If oracle algorithm $\mathcal{A}$ requires multiple (pseudo-)oracles, say oracle $\mathcal{O}_1$ and pseudo-oracle $\mathcal{O}_2$, one should compose the pseudo-oracle last, *i.e.* use the system composition $(\mathcal{A}^{\mathcal{O}_1})^{\mathcal{O}_2}$. By definition, in $\mathcal{A}^{\mathcal{O}_2}$, the pseudo-oracle $\mathcal{O}_2$ *does not have access to* $\mathcal{O}_1$. In many cases, *e.g.* rewinding-based $\mathcal{O}_2$, this means that $\mathcal{A}^{\mathcal{O}_2}$ will not work as expected, *cf.* Remark 4.10.

Definition 4.9 (Black-Box Pseudo-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 (such as its code). We state the following trivial consequence.

Corollary 4.2. *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.*

In this thesis, we only consider pseudo-oracles that fulfill the black-box property.

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.

As composition-order invariance may only hold for protocols with a certain (bounded) round complexity, we first define protocol rounds.

Definition 4.10 (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 4.11 (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

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

Phrased intuitively, 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 $\mathit{view}_{\mathcal{A}}$ only, whereas in $\langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}}$ it has access to $\mathit{view}_{\langle \mathcal{B}, \mathcal{A} \rangle}$ (where, by abuse of notation, we write $\mathit{view}_{\langle \mathcal{B}, \mathcal{A} \rangle}$ for the view of the entity $\langle \mathcal{B}, \mathcal{A} \rangle$ as explained above).

Remark 4.6. Definition 4.11 is quite abstract. It helps to consider the pseudo-oracle $\mathcal{O}_{\mathsf{CCA}}$ from Section 4.3.3. 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 $\mathcal{O}_{\mathsf{CCA}}$ intuitively ensures that, despite this difference, the values extracted by $\mathcal{O}_{\mathsf{CCA}}$ for $\mathcal{A}$ remain unchanged.

Remark 4.7 (Relation to Oracles)**.** Ordinary oracles are evidently black-box and ∞-robust composition-order invariant (w.r.t. unbounded algorithms).

Remark 4.8. We stress that composition-order invariance is a non-trivial property of pseudo-oracles. Indeed, to verify that the committed-value pseudo-oracle $\mathcal{O}_{\mathsf{CCA}}$ satisfies composition-order invariance, we crucially rely on computational assumptions. For that reason, Definition 4.11 restricts to PPT algorithms.

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

Definition 4.12 (k-Robust Quasi-PPT)**.** A black-box pseudo-oracle is *k-robust quasi-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

$$\{\mathrm{out}_{\mathcal{B},\mathcal{A}}\langle \mathcal{B}(x),\, \mathcal{A}^{\mathcal{O}}(y)\rangle(1^\kappa, z)\}_{\kappa\in\mathbb{N},x,y,z\in\{0,1\}^*}$$
$$\overset{s}{\approx} \{\mathrm{out}_{\mathcal{B},\mathcal{S}}\langle \mathcal{B}(x),\, \mathcal{S}(y)\rangle(1^\kappa, z)\}_{\kappa\in\mathbb{N},x,y,z\in\{0,1\}^*}.$$

Terminology 2 (Asymptotics)**.** We say that a pseudo-oracle $\mathcal{O}$ is $O(k)$-robust composition-order-invariant resp. quasi-PPT if $\mathcal{O}$ is f-robust composition-order-invariant resp. quasi-PPT for every $f \in O(k)$.

All properties exist in an enhanced form, where the algorithms are given access to a complexity oracle $\mathcal{O}_{\mathsf{comp}}$.

Definition 4.13 (Enhanced properties)**.** Let $\mathcal{O}$ be a *black-box* pseudo-oracle and $\mathcal{O}_{\mathsf{comp}}$ be a *deterministic stateless* complexity oracle with bookkeeping algorithm $\mathcal{H}_{\mathcal{L}}$. Let $\mathcal{A}$ be a PPT oracle machine which is admissible for $\mathcal{O}_{\mathsf{comp}}$ and let $\mathcal{B}$ be a PPT machine. Let $k = k_{\mathcal{B}} + k_{\mathsf{ass}}$ bound the rounds of interaction $k_{\mathcal{B}}$ between $\mathcal{B}$ and $\mathcal{A}$ *plus* the maximal number k_{ass} of invalidation queries (*i.e.* $|\mathcal{L}|$).

In this situation, we define

- k-robust composition order invariance w.r.t. PPT algorithms (Definition 4.11)
- k-robust quasi-PPT (Definition 4.12)

analogous to the original definitions with $\mathcal{A}^{\mathcal{O}_{\mathsf{comp}}}$ in place of $\mathcal{A}$ (where $\mathcal{H}_{\mathcal{L}}$ always remains an external party interacting with $\mathcal{A}^{\mathcal{O}_{\mathsf{comp}}}$ or $\langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\mathsf{comp}}}\rangle$). The resulting notion $\mathfrak{N}$ is called *enhanced $\mathfrak{N}$ w.r.t. complexity oracle $\mathcal{O}_{\mathsf{comp}}$*.

Remark 4.9. Definition 4.13 uses the stateless interpretation of $\mathcal{O}_{\mathsf{comp}}$ with an explicit left side $\mathcal{H}_{\mathcal{L}}$ because otherwise, invalidation of assumptions could be undone by rewinding, which renders our security notions nonsensical. Consequently, the number of invalidated assumptions becomes part of the robustness parameter k.

Remark 4.10. In Definition 4.13, the resulting machines $(\mathcal{A}^{\mathcal{O}_{\mathsf{comp}}})^{\mathcal{O}}$ and $\langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\mathsf{comp}}}\rangle^{\mathcal{O}}$ are again admissible w.r.t. $\mathcal{O}_{\mathsf{comp}}$. This is non-trivial, but easy to verify. It is a consequence of the *perfect* guarantees of admissibility. If admissibility were non-perfect, *e.g.* allow a negligible probability of violating the bookkeeping guarantee (*i.e.* accessing $\mathcal{O}_{\mathsf{comp}}$ with name $\notin \mathcal{L}$), then standard pathological attacks exploiting rewinding would violate the guarantee with probability 1, and thus not be admissible.

Remark 4.11. The composition order $(\mathcal{A}^{\mathcal{O}_{\mathsf{comp}}})^{\mathcal{O}}$ in Definition 4.12 is a necessity for rewinding-based $\mathcal{O}$: To execute $\mathcal{A}$ in its head, $\mathcal{O}$ has to answer queries of $\mathcal{A}$ to $\mathcal{O}_{\mathsf{comp}}$, which can be hard. (Giving $\mathcal{O}$ direct access to $\mathcal{O}_{\mathsf{comp}}$ empowers $\mathcal{O}$ beyond a mere pseudo-oracle. It is an option, but one must be very careful not to accidentally trivialize security notion. For example, the admissibility of $\mathcal{A}^{\mathcal{O}}$ w.r.t. $\mathcal{O}_{\mathsf{comp}}$ would depend on $\mathcal{O}$ and not be automatic anymore.)

4.3.3. Properties of Commitment Schemes

We require a commitment scheme which is *concurrently extractable* and *long-term trapdoor*[2] to achieve concurrently composable multi-party computation with long-term security (Sections 4.4 and 4.5).

Consequently, we define security notions in the presence of a committed-value oracle $\mathcal{O}_{\mathsf{CCA}}$ (and a complexity oracle $\mathcal{O}_{\mathsf{comp}}$ to model invalidation of assumptions). The security w.r.t. a committed-value oracle will be important, as it allows to concurrently extract adversarially created commitments, while, at the same time, simulating commitments of honest parties. The committed-value oracle $\mathcal{O}_{\mathsf{CCA}}$ for COM plays the receiver of COM in an arbitrary number of sessions. Upon completion of a commit phase in session s, $\mathcal{O}_{\mathsf{CCA}}$ outputs $(\mathtt{End}, s, v_s, \mathit{view}_{\mathsf{R}_s})$. (The view of the receiver is returned for technical reasons. For public-coin COM, it contains no additional information anyway.)

Let $\mathsf{insecure}$ be a family of predicates (for $t \in \mathcal{T}$) such that $\mathsf{insecure}_t(\mathcal{L}) = 1$ if the binding property of commitments of type t is considered not to hold due to the invalidated assumptions $\mathcal{L}$. If $\mathcal{O}_{\mathsf{comp}}$ is a trivial oracle, *e.g.* $\mathcal{O}_{\mathsf{comp}} = \perp$, one recovers the usual notions of binding, *etc.*

Definition 4.14 (CCA-Binding)**.** Let COM be a typed commitment scheme with message space M. Let $\mathcal{O}_{\mathsf{CCA}}$ be a (pseudo-)oracle whose interface is described below. Let $\mathrm{Exp}^{\text{CCA-bind}}_{\mathcal{A},\mathsf{COM},\mathcal{O}_{\mathsf{CCA}}}(\kappa, z)$ 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 $\mathcal{O}_{\mathsf{CCA}}$ provided through the game $\mathcal{G}$. Formally, this is the interaction $\langle \mathcal{G}, \mathcal{A} \rangle^{\mathcal{O}_{\mathsf{CCA}}}$ where the game $\mathcal{G}$ passes messages between $\mathcal{A}$ and $\mathcal{O}_{\mathsf{CCA}}$ as follows:
   - Through $\mathcal{G}$, $\mathcal{O}_{\mathsf{CCA}}$ 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, $\mathcal{O}_{\mathsf{CCA}}$ first generates a fresh setup $\mathrm{ck}_s \leftarrow \mathsf{Setup}(1^\kappa, t)$ and sends it to $\mathcal{A}$ (through $\mathcal{G}$)).

[2]Looking ahead, the commitment scheme we construct is actually *statistically* trapdoor according to the standard definition of trapdoor commitment schemes where no committed-value oracle is provided.

- Whenever a session s is finished, $\mathcal{O}_{\mathsf{CCA}}$ responds with $(\mathtt{End}, s, v_s, \mathit{view}_{\mathsf{R}_s})$, 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 $(\mathtt{End}, s, v_s)$ to $\mathcal{A}$ (but not $\mathit{view}_{\mathsf{R}_s}$).

2. In any session s whose commit phase has finished, the adversary may complete the unveil phase for s. This phase is simulated by the game (which has access to $\mathit{view}_{\mathsf{R}_s}$). Suppose the receiver R_s accepts a decommitment to $v \neq \perp$. If $v \neq v_s$ and $v \in M$, 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.* $\mathcal{O}_{\mathsf{CCA}}$ if for every PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for every $\kappa \in \mathbb{N}$ and every $z \in \{0,1\}^*$, it holds that $\Pr[\mathrm{Exp}^{\text{CCA-bind}}_{\mathcal{A},\mathsf{COM},\mathcal{O}_{\mathsf{CCA}}}(\kappa, z) = 1] \leq \mathsf{negl}(\kappa)$. We then call $\mathcal{O}_{\mathsf{CCA}}$ a *committed-value (pseudo-)oracle.*

Some remarks are in order: First, Definition 4.14 is the binding analogue to *CCA-hiding* of [CLP10]. Second, it is a multi-challenge variant (but as usual, multi- and single-challenge are equivalent by a standard hybrid argument). Third, it would have been more straightforward to have the game or $\mathcal{O}_{\mathsf{CCA}}$ play the honest receivers in both commit and unveil phase. However, we want a committed-value pseudo-oracle with the interface as in Definition 4.14, *i.e.* only the commit phase. Last, $\mathcal{O}_{\mathsf{CCA}}$ rewinds even the game, only strengthening the notion of binding. If $\mathcal{O}_{\mathsf{CCA}}$ is $O(1)$-robust COI, then a game with a constant number of challenges can be handled as a left side, so that $\mathcal{O}_{\mathsf{CCA}}$ does not rewind the game anymore. In contrast to other definitions of CCA oracles, $\mathcal{O}_{\mathsf{CCA}}$ only provides the value committed to, but not the decommitment.

The definition of *enhanced* CCA-binding *w.r.t. a complexity oracle* $\mathcal{O}_{\mathsf{comp}}$ is straightforward. We use stateless $\mathcal{O}_{\mathsf{comp}}$ and let the game handle invalidations.

Definition 4.15 (Enhanced CCA-Binding)**.** Let COM and $\mathcal{O}_{\mathsf{CCA}}$ be as in Definition 4.14. Let $\mathcal{O}_{\mathsf{comp}}$ be a *deterministic stateless* complexity oracle. Define $\mathrm{Exp}^{\text{enh-CCA-bind}}_{\mathcal{A},\mathsf{COM},\mathcal{O}_{\mathsf{comp}},\mathcal{O}_{\mathsf{CCA}}}(\kappa, z)$ to be the output of the following experiment, which is identical to the experiment from Definition 4.14, except for following modifications:

1. The adversary has oracle access to $\mathcal{O}_{\mathsf{comp}}(1^\kappa, z)$.[3]

2. $\mathcal{O}_{\mathsf{CCA}}$: If the commit phase in session s is completed and $\mathsf{insecure}_t(\mathcal{L}) = 1$ for $t = \mathsf{type}(s)$ holds, then output $(\mathtt{End}, s, v_s, \mathit{view}_{\mathsf{R}_s})$ with $v_s = \mathtt{broken}$.

3. The adversary may send $(\mathtt{invalidate}, \mathrm{name})$ to the game, which adds name to the list $\mathcal{L}$ of invalidated assumption.

4. To win in session s, additionally $\mathsf{insecure}_t(\mathcal{L}) = 0$ for $t = \mathsf{type}(s)$ must hold.

[3]$\mathcal{O}_{\mathsf{CCA}}$ has black-box access to $\langle \mathcal{G}, \mathcal{A}^{\mathcal{O}_{\mathsf{comp}}} \rangle$, but no direct access to $\mathcal{O}_{\mathsf{comp}}$, *cf.* Remark 4.10.

We call an adversary *admissible* if it only queries $\mathcal{O}_{\mathsf{comp}}$ on $(\texttt{oracle}, \text{name}, m)$ after it sent $(\texttt{invalidate}, \text{name})$ to the game to invalidate assumption name.

We say COM is *enhanced CCA-binding w.r.t. $\mathcal{O}_{\mathsf{CCA}}$ and $\mathcal{O}_{\mathsf{comp}}$* if for every admissible PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for every $\kappa \in \mathbb{N}$ and every $z \in \{0,1\}^*$, it holds that $\Pr[\text{Exp}^{\text{enh-CCA-bind}}_{\mathcal{A},\mathsf{COM},\mathcal{O}_{\mathsf{comp}},\mathcal{O}_{\mathsf{CCA}}}(\kappa, z) = 1] \leq \mathsf{negl}(\kappa)$.

Terminology 3 (Value Committed To)**.** Let COM be a (typed) commitment scheme with message space M which is (enhanced) CCA-binding w.r.t. committed-value oracle $\mathcal{O}_{\mathsf{CCA}}$ and complexity oracle $\mathcal{O}_{\mathsf{comp}}$. Let $v \in M \cup \{\bot, \bot_{ext}, \texttt{broken}\}$ be the value which is part of the output of $\mathcal{O}_{\mathsf{CCA}}$ in an interaction with a (possibly malicious) committer. If $v \in M$, we say that v is the *value committed to.* If $v \notin M$, *i.e.* $v \in \{\bot, \bot_{ext}, \texttt{broken}\}$, we do not consider the commitment to have a value.

We define now define trapdoor commitment schemes that allow to generate (indistinguishable) dummy commitments that can later be opened to an arbitrary value. Again, we consider a variant where the security holds in the presence of a committed-value oracle $\mathcal{O}_{\mathsf{CCA}}$ and a complexity oracle $\mathcal{O}_{\mathsf{comp}}$.

Definition 4.16 (Enhanced Trapdoor Commitment Scheme)**.** Let $(\mathsf{Setup}, \mathsf{C}, \mathsf{R})$ be a typed commitment scheme with message space M and type space $\mathcal{T}$, and let $(\mathsf{TSetup}, \mathsf{C}_{\text{trap}})$ be algorithms which can be used in place of $(\mathsf{Setup}, \mathsf{C})$. Let $\mathcal{O}_{\mathsf{comp}}$ be a (stateful) complexity oracle and $\mathcal{O}_{\mathsf{CCA}}$ be a committed-value (pseudo-)oracle. Then, $\mathsf{TRAPCOM} = (\mathsf{Setup}, \mathsf{C}, \mathsf{R}, \mathsf{TSetup}, \mathsf{C}_{\text{trap}})$ is called *enhanced trapdoor* w.r.t. $\mathcal{O}_{\mathsf{CCA}}$ and $\mathcal{O}_{\mathsf{comp}}$ if

- $\langle \mathsf{C}, \mathsf{R} \rangle$ and $\langle \mathsf{C}_{\text{trap}}, \mathsf{R} \rangle$ are typed commitment schemes with message space M and types $\mathcal{T}$, and
- for every admissible PPT adversary $\mathcal{A}$, it holds that

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

  that is, the ensembles are statistically indistinguishable.

The experiment (and random variable) $\text{Exp}^{\text{ETD}}_{\mathcal{A},\mathsf{TRAPCOM},\mathcal{O}_{\mathsf{comp}},\mathcal{O}_{\mathsf{CCA}},}(\kappa, b, z)$ is defined as follows:

1. Run $\mathcal{A}^{\mathcal{O}_{\mathsf{comp}}}(1^\kappa, z)$ where $\mathcal{A}^{\mathcal{O}_{\mathsf{comp}}}$ interacts with the game $\mathcal{G}$ as follows.
2. First, $\mathcal{A}^{\mathcal{O}_{\mathsf{comp}}}$ sends $(\texttt{Setup}, t)$. If $b = 0$, set $\text{ck} \leftarrow \mathsf{Setup}(1^\kappa, t)$. Otherwise, set $(\text{ck}, \text{td}) \leftarrow \mathsf{TSetup}(1^\kappa, t)$. The experiment sends ck to $\mathcal{A}$.
3. By sending $(\texttt{Start}, v)$ to $\mathcal{G}$, $\mathcal{A}^{\mathcal{O}_{\mathsf{comp}}}$ starts the commit phase of $\mathsf{TRAPCOM}$, acting as receiver. If $b = 0$, the game $\mathcal{G}$ runs the code of the honest committer C on input $(1^\kappa, \text{ck}, t, v)$. If $b = 1$, the game $\mathcal{G}$ runs the code of the trapdoor committer C_{trap} on input $(1^\kappa, \text{ck}, t, |v|, \text{td})$.

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 $\mathcal{A}^{\mathcal{O}_{\mathsf{comp}}}$ access to a committed-value (pseudo-)oracle $\mathcal{O}_{\mathsf{CCA}}$ as in Definition 4.14. Concretely, we consider $\langle \mathcal{G}, \mathcal{A}^{\mathcal{O}_{\mathsf{comp}}} \rangle^{\mathcal{O}_{\mathsf{CCA}}}$ as the complete experiment.

6. The experiment outputs the view of $\mathcal{A}$.

Again, as in the CCA-binding experiment, $\mathcal{O}_{\mathsf{CCA}}$ rewinds the whole game. This only makes the adversary (and hence security notion) stronger.

Remark 4.12. We require the trapdoor property to hold even if all hardness assumptions in $\mathcal{O}_{\mathsf{comp}}$ are invalidated. The intuition is that honest parties do not know whether a hardness assumption is broken, and thus, for the commitment scheme to protect the committed value, it must be (statistically) hidden even if the assumptions were broken before the commit phase. Perhaps surprisingly, we do not simply consider *unbounded* adversaries. The reason is that we prove composition-order invariance by reduction to enhanced binding. Against unbounded adversaries, our commitments are not binding, and COI may not hold. However, without COI, $\mathcal{O}_{\mathsf{CCA}}$ is not very useful.

Remark 4.13. As we consider PPT adversaries $\mathcal{A}$ but require their views to be statistically indistinguishable, Definition 4.16 captures a *long-term* variant of the trapdoor property (and not a statistical one which holds against unbounded adversaries).

Remark 4.14 (Multi-Challenge Experiments)**.** In the *multi-challenge enhanced binding experiment*, the malicious committer may concurrently engage with arbitrarily many honest receivers (and on different types). It wins if it wins in any session. In the *multi-challenge trapdoor experiment*, the malicious receiver may concurrently engage with arbitrarily many committers (and on different types).

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

4.3.4. Constructions

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

Construction 4 (PRS Commitment Scheme (adapted from [GLP+15]))**.** Let $\kappa \in \mathbb{N}$ be a security parameter. Let $\mathsf{COM}' = (\mathsf{Setup}', \mathsf{C}', \mathsf{R}')$ be a typed commitment scheme with message space $M = \{0,1\}^{\mu(\kappa)}$ for polynomial μ and type space $\mathcal{T}$. Let $\ell = \ell(\kappa)$ denote a round parameter. The (typed) PRS commitment scheme with ℓ rounds and *base commitment* COM' is denoted by PRS_ℓ or just PRS. It has message space M and type space $\mathcal{T}$ and is defined as follows.

Setup. Generate $\mathrm{ck}^b_{i,j} \leftarrow \mathsf{Setup}'(1^\kappa, t)$ for $b \in \{0,1\}$, $i \in [\kappa]$, $j \in [\ell]$.

Commit Phase. On common input $(1^\kappa, t)$ and private input $v \in M$ for C:

1. The committer C chooses $\kappa \cdot \ell$ pairs of random shares $(s^0_{i,j}, s^1_{i,j})$ of v, *i.e.* for every i, j it holds that $s^0_{i,j} \oplus s^1_{i,j} = v$. Then, C and R run COM' to commit to $s^b_{i,j}$ for $b \in \{0,1\}$, $i \in [\kappa]$, $j \in [\ell]$ under commitment key $\mathrm{ck}^b_{i,j}$ and type t (in parallel) to obtain commitments $c^b_{i,j}$.
2. For $j = 1, \ldots, \ell$ sequentially:
   a) The receiver R sends a challenge string $r_j = (r_{1,j}, \ldots, r_{\kappa,j}) \xleftarrow{\$} \{0,1\}^\kappa$.
   b) The committer C unveils the commitments $c^{r_{1,j}}_{1,j}, \ldots, c^{r_{\kappa,j}}_{\kappa,j}$. 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 $u^0_{1,1} \oplus u^1_{1,1} = \cdots = u^0_{\kappa,\ell} \oplus u^1_{\kappa,\ell} = v$, where $u^b_{i,j}$ is the message unveiled for commitment $c^b_{i,j}$.

Terminology 4. In the following, we call the commitment schemes used within the PRS commitment scheme *base commitment schemes*, to distinguish them from the *PRS commitment scheme* itself.

Theorem 4.1. *Let $k \in \mathbb{N}$ be a parameter (which may depend on κ). Let $\mathcal{O}_{\mathsf{comp}}$ be a complexity oracle. Let $\mathsf{COM}'_1, \ldots, \mathsf{COM}'_n$ be base commitment schemes such that for every $t \in \{1, \ldots, n\}$:*

1. *COM'_t has a stateless receiver.*
2. *The commit phase of COM'_t has at most k rounds and the first message is sent by the committer C'_t.*
3. *COM'_t has a non-interactive unveil phase.*
4. *COM'_t is enhanced binding w.r.t. $\mathcal{O}_{\mathsf{comp}}$.*
5. *COM'_t is trapdoor w.r.t. $\mathcal{O}_{\mathsf{comp}}$ with trapdoor committing algorithm $\mathsf{C}'_{\mathrm{trap}_t}$.*

Define a typed commitment scheme COM with types $\{1, \ldots, n\}$ and round parameter $\ell \in \omega(k(\kappa)\log(\kappa))$ as follows:

- ***Inputs:*** *Common input is $(1^\kappa, t)$. Private input to C is v.*
- ***Setup:*** *Setup for PRS_ℓ (i.e. for base commitment COM'_t and ℓ rounds).*
- ***Commit Phase:***
  1. ***PRS commit:*** *Run the (typed) PRS commit phase of PRS_ℓ for type t. Let τ_{prs} be the PRS commitment transcript.*

2. ***Argument of Knowledge (AoK):*** *Run Blum's graph hamiltonicity AoK protocol κ-fold in parallel with base commitments COM'_t to prove: τ_{prs} is a valid PRS commitment to some value $v \in M$.*

- ***Unveil Phase:*** *Run the corresponding PRS unveil phase.*

Let $\mathcal{O}_{\mathsf{CCA}}$ be the following pseudo-oracle, where $\mathcal{A}$ denotes its caller:

- *$\mathcal{O}_{\mathsf{CCA}}$ allows $\mathcal{A}$ to choose common inputs $(1^\kappa, t)$ and interact with an honest receiver R_s in session s in arbitrarily many concurrent sessions. For this, $\mathcal{O}_{\mathsf{CCA}}$ first generates a fresh setup $\mathrm{ck}_s \leftarrow \mathsf{Setup}(1^\kappa, t)$ (per type and session) and sends it to $\mathcal{A}$.*
- *$\mathcal{O}_{\mathsf{CCA}}$ runs the rewinding-based extraction of PRS commitments as in [GLP+15],* cf. *Appendix A.2.1 for more details. Let v_s denote the extracted value (which may be $\bot$) received in (main thread) session s. (If extraction failed, v_s is the special symbol $\bot_{ext}$.)*
- *When the commit phase of session s completes, $\mathcal{O}_{\mathsf{CCA}}$ outputs $(\mathtt{End}, s, v_s, view_{\mathsf{R}_s})$ where v_s is replaced by*
  - *$\bot$ if R rejected (the AoK), or*
  - *$\mathtt{broken}$ if $\mathsf{insecure}_{\mathsf{type}(s)}(\mathcal{L}) = 1$, where $\mathcal{L}$ is the list of invalidated assumptions.*[4]

We only consider adversaries are admissible w.r.t. $\mathcal{O}_{\mathsf{comp}}$ and which invalidate at most $O(k)$ assumptions. Then the following holds for COM w.r.t. $\mathcal{O}_{\mathsf{CCA}}$ and $\mathcal{O}_{\mathsf{comp}}$ and such adversaries:

1. *COM has at most $O(k + \ell)$ rounds.*
2. *COM has a non-interactive unveil phase.*
3. *COM is enhanced CCA-binding w.r.t. $\mathcal{O}_{\mathsf{CCA}}$ and $\mathcal{O}_{\mathsf{comp}}$.*
4. *COM is enhanced trapdoor w.r.t. $\mathcal{O}_{\mathsf{comp}}$ and $\mathcal{O}_{\mathsf{CCA}}$ for some trapdoor committing algorithm $\mathsf{C}_{\mathrm{trap}}$.*
5. *$\mathcal{O}_{\mathsf{CCA}}$ is black-box and $O(k)$-robust composition-order invariant.*
6. *$\mathcal{O}_{\mathsf{CCA}}$ is $O(k)$-robust quasi-PPT.*

Note that in Theorem 4.1, we use a parallel repetition of Blum's graph hamiltonicity protocol instead of a general AoK. This is only for simplicity and to avoid even more *enhanced* definitions.

Proof. The claims in Items 1 and 2 follow immediately.

[4]Note that any (black-box) $\mathcal{O}_{\mathsf{CCA}}$ can trivially reconstruct $\mathcal{L}$, *i.e.* by recomputing all (invalidation) messages $\mathcal{A}$ sent.

Claim 4.1 (Item 3)**.** COM *is enhanced CCA-binding w.r.t.* $\mathcal{O}_{\mathsf{CCA}}$ *and* $\mathcal{O}_{\mathsf{comp}}$.

Proof. This follows immediately from Lemma A.2, the enhanced version of the generalized robust extraction lemma of [GLP$^+$15]. (Namely, an adversary only wins if the extracted value differs from its unveiled value, hence if it violates the validity constraint.) $\square$

In order to prove the trapdoor property, we must first establish the properties of $\mathcal{O}_{\mathsf{CCA}}$.

Claim 4.2 (Item 6)**.** $\mathcal{O}_{\mathsf{CCA}}$ *is black-box and* ∞*-robust quasi-PPT.*

Proof. By definition, $\mathcal{O}_{\mathsf{CCA}}$ only uses the adversary in a black-box way (namely as a next-message function), hence is a black-box pseudo-oracle. Moreover, since $\mathcal{O}_{\mathsf{CCA}}$ is rewinding-based and the rewinding schedule is already implemented in polynomial time, it is ∞-robust quasi-PPT. (Unlike [GLP$^+$15], there is no brute-force oracle as a part of $\mathcal{O}_{\mathsf{CCA}}$.) $\square$

Claim 4.3 (Item 5)**.** $\mathcal{O}_{\mathsf{CCA}}$ *is* $O(k)$*-robust composition-order invariant w.r.t. PPT algorithms.*

Proof sketch. We only give a brief proof sketch, assuming familiarity with PRS preamble extraction via the recursive rewinding schedule defined by $\mathsf{recurse}$, *e.g.* from [PTV14; GLP$^+$15]. For simplicity, we ignore the complexity oracle $\mathcal{O}_{\mathsf{comp}}$. A more detailed proof is given in Appendix A.2.4. For composition-order invariance, we consider a pair $\mathcal{A}$, $\mathcal{B}$ of interacting PPT algorithms, such that $\langle \mathcal{B}, \mathcal{A} \rangle$ is at most k rounds, and show that

$$\begin{aligned} &\{\mathrm{out}_{\mathcal{B},\mathcal{A}} \langle \mathcal{B}(x), \mathcal{A}^{\mathcal{O}_{\mathsf{CCA}}}(y) \rangle (1^\kappa, z)\}_{\kappa\in\mathbb{N},x,y,z\in\{0,1\}^*} \\ \overset{s}{\approx}\ &\{\mathrm{out}_{\mathcal{B},\mathcal{A}} \langle \mathcal{B}(x), \mathcal{A}(y) \rangle^{\mathcal{O}_{\mathsf{CCA}}} (1^\kappa, z)\}_{\kappa\in\mathbb{N},x,y,z\in\{0,1\}^*}. \end{aligned}$$

Note that the rewinding schedule will be different for $\langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\mathsf{CCA}}} \rangle$ and $\langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}_{\mathsf{CCA}}}$, because in the former, $\mathcal{O}_{\mathsf{CCA}}$ cannot rewind the "left side" $\mathcal{B}$, and in the latter, there is no "left side", and $\mathcal{O}_{\mathsf{CCA}}$ will rewind $\mathcal{B}$ and $\mathcal{A}$ together.

By fixing the randomness of $\mathcal{B}$ and $\mathcal{A}$, and assuming suitably partitioned and fixed randomness of $\mathcal{O}_{\mathsf{CCA}}$, one sees that the main thread of execution of $\langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\mathsf{CCA}}} \rangle$ and $\langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}_{\mathsf{CCA}}}$ is identical, until an extracted value v_s of session s differs.[5] Thus, it suffices to bound the probability (over the randomness of $\mathcal{O}_{\mathsf{CCA}}$) of this failure event E. Since the AoK is extractable, it ensures that PRS preambles are well-formed and consistent, the failure event is subsumed by the following failure events:

- PRS extraction in either $\langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\mathsf{CCA}}} \rangle$ or $\langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}_{\mathsf{CCA}}}$ fails. Since $\ell \in \omega(k(\kappa \log(\kappa)))$ by definition of ℓ, this happens with negligible probability by the generalized robust extraction lemma (Lemma A.1).

[5]Note that it suffices to consider the main thread only. Look-ahead threads can differ (and will differ, since the rewinding schedule is different).

- PRS extraction succeeds with value $v_{s,0}$ resp. $v_{s,1}$ but $v_{s,0} \neq v_{s,1}$. This yields two failure events when extracting the respective AoK:[6]
  - AoK extraction fails. This has negligible probability.
  - AoK extraction succeeds. Then the extracted value v'_s of the PRS commitment disagrees with either the extraction in $\langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\mathsf{CCA}}} \rangle$ or $\langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}_{\mathsf{CCA}}}$. As internally, the PRS extractor provides a decommitment for some slot (and the AoK extraction as well), this constitutes a break of the binding property of some COM'_t. Thus, this happens with negligible probability.

Overall, we see that $\Pr[\mathrm{E}] \leq \mathsf{negl}^{\mathsf{PRS}}(\kappa) + \mathsf{negl}^{\mathrm{AoK}}(\kappa) + \mathsf{negl}^{\mathrm{Binding}}(\kappa)$. Thus, except with negligible probability, the view (of the main thread of execution) of $\langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\mathsf{CCA}}} \rangle$ and $\langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}_{\mathsf{CCA}}}$ is identical, and the claim follows.

To deal with $\mathcal{O}_{\mathsf{comp}}$, consider deterministic stateless $\mathcal{O}_{\mathsf{comp}}$ with bookkeeping algorithm $\mathcal{H}_{\mathcal{L}}$ (and (auxiliary) input $(1^\kappa, z)$), and let $\mathcal{H}_{\mathcal{L}}$ be part of the left side (where it is *always* present). Note that at most $O(k)$ assumptions are invalidated, so the left side still has $O(k)$ rounds asymptotically. Thus, an analogous argument shows

$$\mathrm{out}_{\mathcal{B},\mathcal{A}} \langle \mathcal{H}_{\mathcal{L}} \parallel \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\mathsf{CCA}}, \mathcal{O}_{\mathsf{comp}}} \rangle \overset{s}{\approx} \mathrm{out}_{\mathcal{B},\mathcal{A}} \langle \mathcal{H}_{\mathcal{L}}, \langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}_{\mathsf{CCA}}, \mathcal{O}_{\mathsf{comp}}} \rangle$$

for admissible PPT adversaries $\mathcal{A}$, $\mathcal{B}$, where we omitted all inputs for conciseness. □

Finally, we show the enhanced trapdoor property, Item 4. The trapdoor commitment setup is obtained by (in the PRS commitment) replacing Setup' with TSetup' and C' with $\mathsf{C}'_{\mathrm{trap}}$, *i.e.* simply using the trapdoor commitment algorithms of the base commitment COM', in the PRS commitments and the AoK commitments

Claim 4.4 (Item 4)**.** COM *is enhanced trapdoor w.r.t.* $\mathcal{O}_{\mathsf{CCA}}$ *and* $\mathcal{O}_{\mathsf{comp}}$ *for trapdoor commitment algorithm* $\mathsf{C}_{\mathrm{trap}}$*.*

Proof. To prove indistinguishability, we consider the following hybrids.

- Hybrid G_0: Real game, *i.e.* the bit b in $\mathrm{Exp}^{\mathrm{ETD}}_{\mathcal{A},\mathsf{COM},\mathcal{O}_{\mathsf{comp}},\mathcal{O}_{\mathsf{CCA}}}$ is 0.
- Hybrid G_1: Same as G_0, except that the AoK also uses TSetup' and $\mathsf{C}'_{\mathrm{trap}}$ (and trapdoor unveil) instead of the real algorithms.
- Hybrid G_2: Same as G_1, except that the AoK is simulated using the (perfect) special honest-verifier zero-knowledge (SHVZK) simulation for Blum's graph hamiltonicity, by equivocating commitments after the challenge was received.
- Hybrid H_j: Same as $H_0 := G_2$, except that for PRS slots $j' \leq j$, the hybrid executes the left commit phase using trapdoor algorithms TSetup' and $\mathsf{C}'_{\mathrm{trap}}$ (and trapdoor unveil) instead of the real algorithms.

[6]To extract the AoK, one uses that the rewinding-based implementation of $\mathcal{O}_{\mathsf{CCA}}$ leads to a PPT algorithm, *e.g.* $\mathcal{S} = \langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\mathsf{CCA}}} \rangle$, when steps of $\mathcal{O}_{\mathsf{CCA}}$ are included (effectively, this is k-robust quasi-PPT). The AoK extraction is again rewinding-based, but rewinds $\mathcal{S}$. Observe that for extracting the AoK, we can freely rewind the left side $\mathcal{B}$, since AoK extraction happens only in the proof that $\Pr[E]$ is small, but is not part of $\mathcal{O}_{\mathsf{CCA}}$. So extracting the AoK is unrelated with k-robustness properties.

- Hybrid $G_3 \coloneqq H_\ell$: The simulation, *i.e.* the bit b in $\mathrm{Exp}^{\mathrm{ETD}}_{\mathcal{A},\mathsf{COM},\mathcal{O}_{\mathsf{comp}},\mathcal{O}_{\mathsf{CCA}}}$ is 1.

All hybrids output the view of $\mathcal{A}$, denoted by $view_{\mathcal{A}}$. Intuitively, indistinguishability of G_0 and G_1, as well as H_i and H_{i+1} obviously follows from the enhanced trapdoor property (Definition 4.16). However, the presence of the pseudo-oracle $\mathcal{O}_{\mathsf{CCA}}$ that uses rewinding makes the argument non-trivial and we have to rely on the composition-order invariance. We demonstrate the argument for the hybrids H_i; the completely analogous case of $G_0 \overset{s}{\approx} G_1$ is left to the reader.

From H_i to H_{i+1}, the only change is that in PRS slot $i+1$, hybrid H_i uses the real algorithms, whereas H_{i+1} uses the trapdoor algorithms. To argue statistical indistinguishability of the outputs out_i of H_i, we proceed in three steps:

(a) Move all commitments in PRS slot $i+1$ out of the hybrid game and into an external committer on the left; more concretely, use the interface from the enhanced trapdoor experiment.

(b) Switch the (now external) algorithm from real to trapdoor algorithms.

(c) Move the (external) committer on the left back into the hybrid game.

We introduce the reduction between H_i and H_{i+1}: Let $\mathcal{H}_i$ be the experiment H_i with $\mathcal{O}_{\mathsf{CCA}}$ factored out, *i.e.* $H_i = \mathcal{H}_i^{\mathcal{O}_{\mathsf{CCA}}}$, and let $\mathrm{E}_b \coloneqq \mathrm{Exp}^{\mathrm{ETD}}_{\mathcal{H}_i,\mathsf{COM}'^{2\kappa},\mathcal{O}_{\mathsf{comp}},\mathcal{O}_{\mathsf{CCA}}}(1^\kappa, b, z)$ be the ETD experiment from Definition 4.16, where $\mathsf{COM}'^{2\kappa}$ denotes the 2κ-fold parallel composition of COM'. (While $\mathcal{O}_{\mathsf{CCA}}$ is made explicit, $\mathcal{O}_{\mathsf{comp}}$ and $\mathcal{H}_{\mathcal{L}}$ are left implicit for readability.) Finally, let $\mathcal{B}_i$ be defined such that $\mathcal{H}_i = \langle \mathrm{E}_0, \mathcal{B}_i \rangle$, *i.e.* make session i explicit and "move" it into E_0. Also observe that $\mathcal{H}_{i+1} = \langle \mathrm{E}_1, \mathcal{B}_i \rangle$. This gives us

$$H_i = \mathcal{H}_i^{\mathcal{O}_{\mathsf{CCA}}} = \mathrm{out}_{\mathcal{B}_i}\langle \mathrm{E}_0, \mathcal{B}_i \rangle^{\mathcal{O}_{\mathsf{CCA}}} \overset{s}{\approx} \mathrm{out}_{\mathcal{B}_i}\langle \mathrm{E}_0, \mathcal{B}_i^{\mathcal{O}_{\mathsf{CCA}}} \rangle,$$

where the first equality holds by definition, the next equality follows by *black-boxness* of $\mathcal{O}_{\mathsf{CCA}}$ (Corollary 4.2), and the statistical indistinguishability follows from $O(k)$-robust composition-order invariance of $\mathcal{O}_{\mathsf{CCA}}$ (for PPT algorithms). We stress that this indistinguishability is *not* automatic for pseudo-oracles and *requires justification*, given by the composition-order invariance. *e.g.* we write $\mathcal{H}_i = G_i^{\mathcal{O}_{\mathsf{CCA}}}$ instead of $(G_i^{\mathcal{O}_{\mathsf{comp}}})^{\mathcal{O}_{\mathsf{CCA}}}$.

Note that the experiment E_b has, compared to COM', additional rounds of interaction, but it is still in $O(k)$ rounds, so $O(k)$-robust properties of $\mathcal{O}_{\mathsf{CCA}}$ are still applicable.

Next, we find

$$\mathrm{out}_{\mathcal{B}_i}\langle \mathrm{E}_0, \mathcal{B}_i^{\mathcal{O}_{\mathsf{CCA}}} \rangle \overset{s}{\approx} \mathrm{out}_{\mathcal{B}_i}\langle \mathrm{E}_1, \mathcal{B}_i^{\mathcal{O}_{\mathsf{CCA}}} \rangle,$$

by first using the k-robust quasi-PPT property of $\mathcal{O}_{\mathsf{CCA}}$ established in Claim 4.2 to replace $\langle \mathrm{E}_0, \mathcal{B}^{\mathcal{O}_{\mathsf{CCA}}} \rangle$ by $\langle \mathrm{E}_0, \mathcal{S}' \rangle$. $\mathcal{S}'$ is a PPT adversary in the enhanced trapdoor game w.r.t. $\mathcal{O}_{\mathsf{comp}}$ (with many commitments in parallel). Using that COM' is an enhanced trapdoor commitment scheme[7] by assumption (and applying another hybrid argument

[7]It is easy to see that a 2κ-fold parallel composition of COM' is still a trapdoor commitment (w.r.t. $\mathcal{O}_{\mathsf{CCA}}$).

over the commitment instances), the switch from the left side E_0 to E_1 is statistically indistinguishable.

Now, we reverse the previous steps, with the same arguments but E_1 as left side to find

$$\mathrm{out}_{\mathcal{B}_i}\langle \mathrm{E}_1, \mathcal{B}_i^{\mathcal{O}_{\mathsf{CCA}}}\rangle \overset{s}{\approx} \mathrm{out}_{\mathcal{B}_i}\langle \mathrm{E}_1, \mathcal{B}_i\rangle^{\mathcal{O}_{\mathsf{CCA}}} = \mathcal{H}_{i+1}^{\mathcal{O}_{\mathsf{CCA}}} = H_{i+1}.$$

Thus, we have shown that $H_0 \overset{s}{\approx} H_\ell$, via uniform reductions $H_i \overset{s}{\approx} H_{i+1}$ (for $0 \le i \le \ell - 1$). As noted, $G_0 \overset{s}{\approx} G_1$ follows completely analogously.

It remains to show that $G_1 \overset{s}{\approx} G_2$, *i.e.* that we can simulate the AoK. To do so, instead of unveiling the real values (using the algorithm of the trapdoor committer), we instead use the special honest-verifier zero-knowledge (SHVZK) simulator for Blum's graph hamiltonicity protocol to generate the messages to unveil. Since the SHVZK simulation is perfect (given perfectly hiding commitments), for any challenge the distribution of unveiled messages is identical for real and simulated commitments. Thus, we can let $\mathsf{C}_{\mathrm{trap}}$ unveil the simulation, instead of the real messages. Again, due to the presence of the pseudo-oracle (which may rewind the interaction, breaking the perfect indistinguishability), we formally argue this by first moving COM' to the left side, doing the switch, and then moving it back to the right side. This is done completely analogous to the hybrid steps. This finishes the proof of Claim 4.4. □

With this, the proof of Theorem 4.1 is complete. □

The base commitment COM' in Theorem 4.1 can be instantiated using a number of assumptions. We note that COM' is a stand-alone-secure commitment scheme, in particular, it need not be universally composable

Proposition 4.1 (Possible Instantiations). *Under the RSA assumption, the DLOG assumption and the SIS assumption, there exist commitment schemes $\mathsf{COM}'_{\mathrm{RSA}}$ [HW09], $\mathsf{COM}'_{\mathrm{DLOG}}$ [P92] and $\mathsf{COM}'_{\mathrm{SIS}}$ [GVW15] in the CRS-hybrid model with*

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

4.4. Security Notions

With UC security and its variants, all entities keep their runtime complexity throughout the whole execution, making them unsuitable to analyze the security of protocols in a setting where cryptographic hardness assumptions may become invalid during the execution.

We extend the established notions of Universal Composability [C01] and long-term security [MU10] to capture a setting where (some) hardness assumptions may adaptively become invalid throughout or after a protocol execution. To this end, we take a route similar to [PS04; CLP10] and provide environment and adversary with a (possibly inefficient) entity called the *helper* $\mathcal{H}$. This helper may provide the (PPT) environment and adversary with complexity oracles that can be used to solve certain problems which are assumed to be hard otherwise. Formally, our notion is cast in the Generalized UC framework [CDPW07], 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 Section 2.4.

Invalidating Hardness Assumptions. To model that hardness assumptions become invalid, the helper $\mathcal{H}$ may provide several deterministic and stateless *complexity oracles* (Section 4.3.1). For example, $\mathcal{H}$ may provide an oracle that solves instances of the DLOG problem. Even in the presence of such an oracle, some cryptographic assumptions are *believed* (and often assumed) to retain their hardness (*e.g.* post-quantum assumptions), while others are believed to become invalid (*e.g.* the RSA assumption).

For assumptions A and B, where B is assumed to be hard relative to A, the security of protocols using B (and A) in the presence of an oracle breaking A can be analyzed within our framework. Typically, such a setting can be encountered when A is an "older" assumption used in a protocol, where it is "updated" to a "newer" assumption B.

More generally, $\mathcal{H}$ could also provide oracles capturing whole complexity classes, *e.g.* via a BQP or EXP oracle.[8]

Our notion can thus be seen as a generalization of everlasting or long-term security [MU10]: Instead of *all* hardness assumptions eventually becoming invalid, we model a setting where *some* assumptions may be come invalid even while the protocol under analysis is executed and other hardness assumptions still exist.

In this chapter, we only consider protocols where the honest parties are classical and run in polynomial time. The helper $\mathcal{H}$ is intentionally not available for honest parties, as they do not need it for protocol execution and any use of $\mathcal{H}$ would constitute a deviation from the honest execution.

At the beginning of the execution, all complexity oracles provided by $\mathcal{H}$ are disabled. This models an execution where all (assumed) hardness assumptions against PPT adversaries are still valid. Throughout the execution, the environment may *invalidate* assumptions by enabling the corresponding complexity oracle. Whenever the environment invalidates an assumption, the adversary is notified, but protocol parties remain oblivious.

Extracting (Statistically Hiding) Commitments. Apart from complexity oracles dealing with (invalidated) hardness assumptions, $\mathcal{H}$ also provides a (pseudo-)oracle that allows the extraction of commitments (*cf.* Section 4.3.3), similar to a CCA oracle. This part is analogous to the helper of [CLP10], with the following differences: The

[8]Note that the entities accessing $\mathcal{H}$ are classical and polynomially bounded. Thus, they can only provide and receive classical inputs resp. outputs of polynomial length, even if the oracle would accept longer or non-classical inputs, *e.g.* quantum states.

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 [GLP+15] to our setting, via pseudo-oracles and a suitably adapted analysis in Section 4.3 and Appendix A.2. 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, due to the presence of the helper $\mathcal{H}$.

We also have to deal with the case that the assumptions requiring for the extractability become invalid during the extraction, making extraction impossible. The committed-value (pseudo-)oracle is thus aware of invalidated assumptions and may abort the extraction.

In a setting where we want to *update* commitments, the helper may not only be parameterized with a single commitment scheme, but possibly with multiple ones. Typically, different schemes may be associated with different hardness assumptions that are assumed to be independent. Thus, a commitment scheme COM_b may remain extractable even if COM_a has lost its extractability.

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

As the commitment key is generated honestly, it is guaranteed to be independent of 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 using ck′ (if the commitment scheme is secure in the presence of $\mathcal{O}_{\mathsf{CCA}}$). 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 keys).

Consistent with the setting of Section 4.3, we consider *typed commitment schemes* (Definition 4.2) in the following. To establish meaningful properties, we require the typed commitment scheme to feature a committed-value (pseudo-)oracle which is *black-box* (Definition 4.9) and *enhanced k-robust composition-order invariant* and *k-robust quasi-PPT* (Definitions 4.11 and 4.12). This allows to a) import protocols with appropriate round complexity into our framework without loss of security due to the committed-value (pseudo-)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 (pseudo-)oracle without having to rewind the challenger in a reduction. Moreover, we say that $\mathcal{H}$ is black-box if all of its (pseudo-)oracles are black-box.

The helper $\mathcal{H}$ is formally defined in Figure 4.1. For the sake of an easier definition and notion, we assume that all (pseudo-)oracles (corresponding to used and possibly invalid hardness assumptions) as well as commitment schemes are fixed and cannot be extended dynamically throughout the execution. However, $\mathcal{H}$ can be easily modified to capture an adaptive setting.

In the following, we only consider helpers with (pseudo-)oracles that have appropriate properties and are, in particular, black-box.

Remark 4.15. The helper $\mathcal{H}$ in Figure 4.1 is parameterized with a (typed) commitment scheme COM and contains a committed-value (pseudo-)oracle $\mathcal{O}_{\mathsf{CCA}}$ for COM. For analyzing the security of protocols in a setting where cryptographic hardness assumptions may become invalid, inclusion of $\mathcal{O}_{\mathsf{CCA}}$ is not necessary *per se*. However, by having $\mathcal{H}$ provide $\mathcal{O}_{\mathsf{CCA}}$, we can construct protocols with strong (composable) security properties from natural setups, which would have been impossible otherwise in a setting where cryptographic hardness assumptions may become invalid.

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; CDPW07; MU10] and discuss the properties of the new notion.

4.4.1. Changes to the Framework.

Due to the technicalities of pseudo-oracles outlined in Section 4.3.2, 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) that may be affected by the rewinding.

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 $\mathcal{O}_{\mathsf{CCA}}$ with the necessary view(s), *cf.* Definition 4.11. 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 $(1^\kappa, z)$.

Helper $\mathcal{H}$

$\mathcal{H}$ is parameterized with

- a security parameter $\kappa \in \mathbb{N}$,
- auxiliary input $z \in \{0,1\}^*$,
- a set of tuples $\mathcal{P}$ where each tuple is of the form $(\text{name}, \mathcal{O}_{\text{name}})$, where $\mathcal{O}_{\text{name}}$ is a deterministic stateless oracle,
- a typed commitment scheme $\mathsf{COM} = (\mathsf{COM}_1, \ldots, \mathsf{COM}_n)$ with committed-value (pseudo-)oracle $\mathcal{O}_{\mathsf{CCA}}$. Each COM_i is associated with a list of assumptions $\mathcal{L}_i$.

Complexity Oracles.
Initially, set $\mathcal{L} = \emptyset$.

- Upon receiving $(\texttt{query})$ from some ideal functionality $\mathcal{F}$, answer with $\mathcal{L}$.
- Upon receiving $(\texttt{invalidate}, \text{name})$ from the environment such that there exists a tuple $(\text{name}, \mathcal{O}_{\text{name}})$ in $\mathcal{P}$, add name to $\mathcal{L}$ and send $(\texttt{invalidated}, \text{name})$ to the adversary.
- Upon receiving $(\texttt{oracle}, \text{name}, m)$ from the environment or the adversary, send $(1^\kappa, z, m)$ to $\mathcal{O}_{\text{name}}$ and send the answer to the environment or the adversary. If $\mathcal{O}_{\text{name}}$ does not exist or name $\notin \mathcal{L}$, return $\perp$.

Committed-Value Oracle.

- Upon receiving an input $(\texttt{corrupt}, P_i, \mathit{sid})$ from the environment, record $(\texttt{corrupt}, P_i, \mathit{sid})$.
- Upon receiving an input $(\texttt{ext-init}, P_i, \mathit{sid}, k, l)$ from a corrupted party P_i in the protocol with SID sid: If $\mathcal{L}_l \cap \mathcal{L} \neq \emptyset$, ignore this message. If there is a recorded session (P_i, sid, k), ignore this message. Otherwise, initialize the k-th sub-session of (P_i, sid) and type l with $\mathcal{O}_{\mathsf{CCA}}$ and receive a commitment key ck. Record session (P_i, sid, k) and return $(\texttt{setup}, \mathit{sid}, k, \text{ck})$ to P_i.
- Upon receiving an input $(\texttt{ext-mesg}, P_i, \mathit{sid}, k, m)$ 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. If $\mathcal{L}_l \cap \mathcal{L} \neq \emptyset$, also ignore this message. Otherwise, send (sid, k, m) to $\mathcal{O}_{\mathsf{CCA}}$ and possibly obtain a reply m'. If m' is a special message $(\texttt{End}, s, v_s, \mathit{view}_{\mathsf{R}_s})$, return $(\texttt{ext-val}, P_i, \mathit{sid}, k, v_s)$ to P_i. Otherwise, return $(\texttt{ext-mesg}, P_i, \mathit{sid}, k, m')$ to P_i.

Figure 4.1.: The helper $\mathcal{H}$.

2. On invocation, $\mathcal{E}$ immediately invokes $\mathcal{H}$, giving its code[9] 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 $\mathsf{Exec}(\pi, \mathcal{A}, \mathcal{Z})(\kappa, z)$ 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 this modification the *Updatable UC framework*.

4.4.2. Protocol Emulation

We continue with the definition of protocol emulation, adapted from [CLP10].

Definition 4.17 (Updatable UC Protocol Emulation)**.** Let π and ϕ be PPT protocols and let $\mathcal{H}$ be the helper of Figure 4.1. We say that π *Updatable-UC-emulates* ϕ if for every PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for every $\mathcal{H}$-aided[10] balanced PPT environment $\mathcal{Z}$, there exists a negligible function negl such that for every $\kappa \in \mathbb{N}, z \in \{0,1\}^*$ it holds that

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

If π emulates ϕ in the Updatable UC framework under this notion of protocol emulation, we write $\pi \geq_{\text{Upd-UC}} \phi$. If π emulates IDEAL($\mathcal{F}$), *i.e.* the ideal protocol of a functionality $\mathcal{F}$, then we say that π Updatable-UC-realizes $\mathcal{F}$. We call the resulting security notion *Updatable UC security.*

We also define a notion of protocol emulation where the environment *never* sends `invalidate` messages to $\mathcal{H}$. Intuitively, this captures a setting where (rewinding-based) extraction of commitments is possible via $\mathcal{H}$, but hardness assumptions remain valid all time. In Berger *et al.* [BBK$^+$23], this is modeled via a helper that is very similar to the one in Figure 4.1, but does not contain any complexity oracles. In line with [BBK$^+$23], we call this notion *UC security with Rewinding* or *Rewinding UC security.* It is easy to see that both notions are equivalent.

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

[10]We restate the definition of $\mathcal{H}$-aided environments due to Canetti, Lin, and Pass [CLP16] with minor syntactic modifications: 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.

Definition 4.18 (Rewinding UC Protocol Emulation)**.** Let π and ϕ be PPT protocols and let $\mathcal{H}$ be the helper of Figure 4.1. We say that π *Rewinding-UC-emulates* ϕ if for every PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for every $\mathcal{H}$-aided balanced PPT environment $\mathcal{Z}$ that never sends `invalidate` to $\mathcal{H}$, there exists a negligible function negl such that for every $\kappa \in \mathbb{N}, z \in \{0,1\}^*$ it holds that

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

If π Rewinding-UC-emulates ϕ, we write $\pi \geq_{\text{R-UC}} \phi$. If π emulates IDEAL($\mathcal{F}$), *i.e.* the ideal protocol of a functionality $\mathcal{F}$, then we say that π Rewinding-UC-realizes $\mathcal{F}$.

Long-Term Protocol Emulation. Often, we want to analyze the security of protocols that provide long-term or even statistical security for at least a subset of parties while using hardness assumptions. For this setting, Definition 4.17 is not appropriate, as the distinguisher (*i.e.* the environment) may not be powerful enough, depending on which oracles $\mathcal{H}$ provides.

Thus, we also define long-term protocol emulation in our framework in analogy to the established definition due to Müller-Quade and Unruh [MU10]. In contrast to Definition 4.17, 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 there are no hard problems whatsoever anymore 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 4.19 (Long-Term Updatable UC Protocol Emulation)**.** Let π and ϕ be PPT protocols and let $\mathcal{H}$ be the helper of Figure 4.1. We say that π *long-term-Updatable-UC-emulates* ϕ if for every PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for every $\mathcal{H}$-aided balanced PPT environment $\mathcal{Z}$, the ensembles $\{\mathsf{ExecS}(\pi, \mathcal{A}, \mathcal{Z})(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^*}$ and $\{\mathsf{ExecS}(\phi, \mathcal{S}, \mathcal{Z})(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^*}$ are statistically indistinguishable.

If π long-term-Updatable-UC-emulates ϕ, we write $\pi \geq_{\text{LT-Upd-UC}} \phi$.

Remark 4.16. In contrast to the definition of long-term security in [MU10], the environment in Definition 4.19 has access to the helper $\mathcal{H}$, enabling it to invalidate (some) hardness assumptions during protocol execution. Additionally, the helper provides a committed-value (pseudo-)oracle, which also does not exist in the original definition.

Often, we are interested in a setting where the environment a) has access to the committed-value (pseudo-)oracle (to circumvent the impossibility results of [MU10]) but b) never invalidates hardness assumptions during the protocol execution. This setting is comparable to the original definition of long-term security, with the addition of the committed-value (pseudo-)oracle. We state the following notion of protocol emulation that captures this setting.

Definition 4.20 (Long-Term Rewinding UC Protocol Emulation)**.** Let π and ϕ be PPT protocols and let $\mathcal{H}$ be the helper of Figure 4.1. We say that π *long-term-Rewinding-UC-emulates* ϕ if for every PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for every $\mathcal{H}$-aided balanced PPT environment $\mathcal{Z}$ that never sends `invalidate` to $\mathcal{H}$, the ensembles $\{\mathsf{ExecS}(\pi, \mathcal{A}, \mathcal{Z})(\kappa, z)\}_{\kappa\in\mathbb{N}, z\in\{0,1\}^*}$ and $\{\mathsf{ExecS}(\phi, \mathcal{S}, \mathcal{Z})(\kappa, z)\}_{\kappa\in\mathbb{N}, z\in\{0,1\}^*}$ are statistically indistinguishable.

If π long-term-Rewinding-UC-emulates ϕ, we write $\pi \geq_{\text{LT-R-UC}} \phi$. Again, it is easy to see that Definition 4.20 is equivalent to the one proposed in [BBK+23].

As (long-term-) Rewinding UC security is a special case of (long-term-) Updatable UC security, (long-term-) Updatable UC security implies (long-term-) Rewinding UC security.

Proposition 4.2. *Let π and ϕ be PPT protocols. If π (long-term-)Updatable-UC-emulates ϕ, then π (long-term-)Rewinding-UC-emulates ϕ.*

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

Proposition 4.3 (Long-term Emulation Implies Classical Emulation)**.** *Let π and ϕ be PPT protocols. If $\pi \geq_{\text{LT-Upd-UC}} \phi$ resp. $\pi \geq_{\text{LT-R-UC}} \phi$ and $\mathcal{H}$ is black-box, then $\pi \geq_{\text{Upd-UC}} \phi$ resp. $\pi \geq_{\text{R-UC}} \phi$ (for the same helper $\mathcal{H}$).*

The proof of Proposition 4.3 is simple and we omit it.

4.4.3. Properties of Our Notions

We now discuss the properties of our notions, which are mostly similar to the properties of (G)UC resp. long-term security.

Proposition 4.4 (Completeness of the Dummy Adversary)**.** *If $\mathcal{H}$ is black-box, then the dummy adversary is complete.*

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

Also, our notions are transitive.

Proposition 4.5 (Transitivity)**.** *Let π_1, π_2, π_3 be PPT protocols. If π_1 (long-term-) Updatable-UC-emulates π_2 and π_2 (long-term-) Updatable-UC-emulates π_3 (for the same black-box helper $\mathcal{H}$), then π_1 (long-term-) Updatable-UC-emulates π_3.*

As the proof is very similar to the one for transitivity of GUC security, we omit it. It is easy to see that (long-term-) Rewinding UC security is transitive, too.

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

Theorem 4.2 (Composition Theorem (based on [CDPW07]))**.** *Let $\mathcal{H}$ be a helper with a black-box committed-value (pseudo-)oracle. Let ρ, π, ϕ be PPT protocols where π and ϕ are $\mathcal{G}$-subroutine-respecting for some global functionality $\mathcal{G}$. If π (long-term-) Updatable-UC-emulates ϕ, then, $\rho^{\phi\to\pi}$ (long-term-) Updatable-UC-emulates ρ.*

It is easy to see that (long-term-) Rewinding UC security is also closed under universal composition.

Proof sketch. We give a proof sketch of Theorem 4.2. To this end, we revisit parts of the proof of the GUC composition theorem in [CDPW06, Proof of Theorem 2.3] pertaining to the helper.

The overall idea of the proof of the GUC composition theorem is as follows. Suppose that there is an environment $\mathcal{Z}$ (possibly with access to global functionalities) that can distinguish between an execution of ρ and $\rho^{\phi \to \pi}$. Then, we can construct an environment $\mathcal{Z}_\pi$ that can distinguish between an execution of π and $\mathcal{D}$ and an execution of ϕ and a simulator $\mathcal{S}_\pi$ with the same non-negligible probability.

To this end, $\mathcal{Z}_\pi$ internally emulates $\mathcal{Z}$, (possibly several instances of) ρ and a special adversary $\hat{\mathcal{A}}_\rho$ and externally interacts with either (possibly several instances of) π and the dummy adversary $\mathcal{D}$ or with (possibly several instances of) ϕ and a simulator $\mathcal{S}_\pi$ for π and $\mathcal{D}$, as well as the global functionalities. All messages are routed such that $\mathcal{Z}$ is presented an execution of ρ or $\rho^{\phi \to \pi}$ and the global functionalities. In particular, $\hat{\mathcal{A}}_\rho$ acts as the adversary for $\mathcal{Z}$. Messages pertaining to sessions of ϕ that are sub-sessions of instances of ρ are forwarded to the external adversary. Messages for other protocols (*e.g.* for ρ) are handled like by the dummy adversary. It remains to be shown that $\mathcal{Z}$'s view when emulated by $\mathcal{Z}_\pi$ is identically distributed as in a "real" execution with $\rho^{\phi \to \pi}$ and $\mathcal{D}$ resp. ρ and $\mathcal{S}$ and the global ideal functionalities. In the case of standard GUC security, this is easy to establish.

In our case, we additionally need to argue that this proof still goes through in the presence of the helper $\mathcal{H}$, similar to the proof of the composition theorem in [CLP10].

Like in [CLP10], it is necessary to show that

1. calls from $\mathcal{Z}$ to $\mathcal{H}$ can be correctly handled by $\mathcal{Z}_\pi$ and
2. calls from internally emulated corrupted parties (*e.g.* from ρ or from "external" sessions invoked by $\mathcal{Z}$) to $\mathcal{H}$ can also be correctly handled by $\mathcal{Z}_\pi$.

Additionally, we need to prove that the distribution of the helper's responses in an execution with $\mathcal{Z}_\pi$ resp. $\mathcal{Z}$ (as described above) does not change.

To this end, we distinguish between the following cases:

- When $\mathcal{Z}$ wants to send $(\texttt{corrupt}, P_i, sid)$ to $\mathcal{H}$:
    1. If the party with identity (P_i, sid) does not exist, invoke it. This may involve the invocation of an "external" session of the protocol of (P_i, sid). Subsequently, corrupt the party (P_i, sid).
    2. Send $(\texttt{corrupt}, P_i, sid)$ to $\mathcal{H}$.
- When an internally emulated corrupted party (P_i, sid) wants to send $(\texttt{ext-init}, P_i, sid, k)$ or $(\texttt{ext-mesg}, P_i, sid, k, m)$ to $\mathcal{H}$, forward this message to $\mathcal{H}$ on behalf of (P_i, sid) (which has been invoked in the previous step).

- Appropriately forward messages from $\mathcal{H}$ for a party (P_i, sid) ($\mathcal{Z}_\pi$ only sees these messages if (P_i, sid) is an internally emulated corrupted party which $\mathcal{Z}_\pi$ also has invoked externally) to the internally emulated (corrupted) party (P_i, sid).
- Messages between $\mathcal{Z}$ and $\mathcal{H}$ pertaining to complexity oracles are simply forwarded.

It follows that the views of the internally emulated environment $\mathcal{Z}$ and the internally emulated corrupted parties remain correctly distributed if the answers from $\mathcal{H}$ remain identically distributed. This follows from the black-box property of the pseudo-oracle $\mathcal{O}_{\mathsf{CCA}}$ within $\mathcal{H}$, which we assume to be black-box (see Definition 4.9).

Also, it is easy to see that if $\mathcal{Z}$ is $\mathcal{H}$-aided (which holds by assumption), then so is $\mathcal{Z}_\pi$. □

Remark 4.17. The proof of the UC composition theorem uses a hybrid argument to replace sessions of ϕ step by step with sessions of π. To this end, it is necessary that the environment $\mathcal{Z}$ can also incorporate the simulator $\mathcal{S}_\pi$. While the proof of the GUC composition theorem is done without a hybrid argument (all sessions of ϕ resp. π can be replaced in a single step as there may be multiple challenge sessions), we stress that environments in our notion *can* internally execute simulators accessing $\mathcal{H}$. The argument is similar to the above.

Remark 4.18. For the above proof sketch, we only require the pseudo-oracle to be *black-box*, but *do not* need additional properties such as the composition-order invariance. To this end, we recall that we have re-defined the execution experiment in Section 4.4.1. As such, switching from the execution with $\mathcal{Z}$ to the execution with $\mathcal{Z}_\pi$ does not involve the creation of any new "real" machines, as the whole execution experiment (with the exception of $\mathcal{H}$) is emulated on a single machine. Thus, $\mathcal{O}_{\mathsf{CCA}}$ always interacts with exactly one machine, without any changes in the order of composition. (However, the composition-order invariance may be needed to prove the security of a protocol in the first place.)

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 without loss of security.

Let π and ϕ be PPT protocols such that $\pi \geq_{\mathrm{UC}} \phi$. 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 $\pi \geq_{\mathrm{UC}} \phi$ implies that $\pi \geq_{\text{Upd-UC}} \phi$. This is because an Updatable UC environment is more powerful than a normal UC environment due to the access to $\mathcal{H}$: First, the complexity oracles of $\mathcal{H}$ could break computational assumptions of π or ϕ, making an indistinguishable simulation impossible. Second, the committed-value oracle of $\mathcal{H}$ could also invalidate assumptions made in the security proof.

Nevertheless, we can show the compatibility of Updatable UC security with UC security for large classes of protocols, namely those whose i) hardness assumptions are not affected by the complexity oracles of $\mathcal{H}$ and ii) have less than or equal to k rounds if the committed-value (pseudo-)oracle provided by $\mathcal{H}$ is black-box (Definition 4.9), enhanced k-robust composition-order invariant (Definitions 4.11 and 4.13) and enhanced k-robust quasi-PPT (Definitions 4.12 and 4.13). In particular, the first criterion is fulfilled by information-theoretically secure protocols. The second criterion is essentially the same as in [CLP10], except for the additional requirements due to the (pseudo-)oracle $\mathcal{O}_{\mathsf{CCA}}$ within $\mathcal{H}$.

Before stating the theorem, we give a formal definition of k-round protocols. As the model of execution in Updatable UC is different from the model of stand-alone execution, we cannot simply reuse the stand-alone definition of k-round protocols (Definition 4.10).

As we eventually want to use the k-robust composition-order invariance (Definition 4.11), we need a definition of k-round protocols within Updatable 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.[11] We thus provide a possible definition here. We stress that any definition that works for [CLP10] works for our setting and vice versa.

Definition 4.21 (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. Input and subroutine output between (dummy) parties and ideal functionalities.
4. Messages between (sub-)parties of π and the adversary.
5. Messages between the adversary and an ideal functionality.

Remark 4.19. Unless governed by Item 3, "immediate" communication between honest (sub)parties 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 (even if a party is corrupted).

With a definition of protocol rounds at hand, we state the following definition of k-round protocols.

Definition 4.22 (k-Round Protocol)**.** Let π be a subroutine-respecting PPT protocol. We say that π is a *k-round protocol* if for every environment $\mathcal{Z}$ and every adversary $\mathcal{A}$ 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,

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

- 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 4.21,
- for each (honest) main party P_i of π (and its sub-parties) a bound $n_i^{\mathcal{A}}$ for the number of rounds for communication with the adversary according to Item 4 in Definition 4.21,
- a bound $n_{\mathcal{A}}$ for the communication of an ideal functionality $\mathcal{F}$ with the adversary according to Item 5 in Definition 4.21 if π is the ideal protocol of $\mathcal{F}$ or IDEAL($\mathcal{F}$) is a subroutine of π.

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

Remark 4.20. Definitions 4.21 and 4.22 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 (at most) 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 4.21. 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 this chapter, 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 4.1. We analyze the round complexity of several protocols, subject to the conditions of Remark 4.21:

- 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 $\mathcal{F}_{\text{COM}}$ and the adversary (*i.e.* the delayed outputs).
- IDEAL($\mathcal{F}_{\text{AUTH}}$) where the message is delivered has four rounds:
  - Two rounds for input and 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.
  - Two rounds for input and subroutine output 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 4.22, 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.* IDEAL($\mathcal{F}_{\text{COM}}$) or IDEAL($\mathcal{F}_{\text{AUTH}}$).

Remark 4.22. 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 4.23. 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 $O(m)$ rounds in π for every (parallel) round of ϕ.

If the number of instances m executed in parallel is known at the time of invocation of ϕ, sender and receiver have the same party ID in all instances and are sub-parties of the same party of π, 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 4.3 (UC Compatibility)**.** *Let ϕ be a subroutine-respecting PPT protocol and let π be a subroutine-respecting PPT protocol with k_1 rounds according to Definition 4.22. Let $\mathcal{H}$ be the helper that is parameterized with a constant number of complexity oracles k_2 and typed commitment scheme* COM *that features a black-box enhanced $O(k)$-robust composition-order invariant and enhanced $O(k)$-robust quasi-PPT committed-value (pseudo-)oracle $\mathcal{O}_{\mathsf{CCA}}$, where $k = k_1 + k_2$ and $k \in O(\mathsf{poly}(\kappa))$. If*

- $\pi \geq_{\text{Stat-UC}} \phi$*, then,* $\pi \geq_{\text{LT-Upd-UC}} \phi$.

- *$\pi \geq_{\text{LT-UC}} \phi$, then $\pi \geq_{\text{LT-R-UC}} \phi$.*
- *$\pi \geq_{\text{UC}} \phi$, then $\pi \geq_{\text{R-UC}} \phi$.*
- *$\pi \geq_{\text{UC}} \phi$ for environments with access to a complexity oracle $\mathcal{O}_{\mathsf{comp}}$ that captures the complexity oracles of $\mathcal{H}$, then $\pi \geq_{\text{Upd-UC}} \phi$.*

Here, $\geq_{\text{UC}}$ denotes (computational) UC emulation, $\geq_{\text{Stat-UC}}$ denotes statistical UC emulation and $\geq_{\text{LT-UC}}$ denotes long-term UC emulation.

Proof. We only prove the first part of Theorem 4.3, as the other parts are very similar. Let π be a subroutine-respecting PPT protocol with up to k_1 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 Updatable 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 UC security, the runtime of $\mathcal{S}$ is polynomial in the runtime 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 a UC execution of π and $\mathcal{D}$ (for messages concerning π) is bounded by $O(k_1)$ if π is a k_1-round protocol according to Definition 4.22.

As π statistically UC-emulates ϕ, this also holds in the UC execution with ϕ and the simulator $\mathcal{S}$ for the dummy adversary. Moreover, the number of `invalidate` queries sent to $\mathcal{H}$ for any Updatable UC environment is bounded by k_2, where k_2 is the number of complexity oracles within $\mathcal{H}$.

For the sake of contradiction, assume that $\pi \not\geq_{\text{LT-Upd-UC}} \phi$ for the dummy adversary and simulator $\mathcal{S}'$ and some Updatable UC environment $\mathcal{Z}$. Let D be an (unbounded) distinguisher that distinguishes with non-negligible advantage, given the output of the Updatable UC environment.

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

First, we split the helper $\mathcal{H}$ into two interactive Turing machines (ITMs) $\mathcal{H}_1$ and $\mathcal{H}_2$, where $\mathcal{H}_1$ is responsible for the complexity oracles of $\mathcal{H}$ and $\mathcal{H}_2$ is responsible for the committed-value (pseudo-)oracle.

Let $\mathcal{E}_1$ be an ITM that is identical to $\mathcal{E}$ as defined in Section 4.4.1, but interacts with $\mathcal{H}_1$ and $\mathcal{H}_2$ instead of $\mathcal{H}$. $\mathcal{E}_1$ also informs $\mathcal{H}_2$ of invalidated assumptions and only allows queries to $\mathcal{O}_{\mathsf{comp}}$ through $\mathcal{H}_1$ if an appropriate `invalidate` message has been previously sent. Clearly, $\mathcal{E}_1$ is PPT and using the black-box property of $\mathcal{O}_{\mathsf{CCA}}$ (within $\mathcal{H}$ resp. $\mathcal{H}_2$), we obtain that the outputs of $\mathcal{E}$ and $\mathcal{E}_1$ are identically distributed.

Let $\mathcal{O}_{\mathsf{comp}}$ be the complexity oracle associated with $\mathcal{H}_1$ and let $\mathcal{O}_{\mathsf{CCA}}$ be the committed-value (pseudo-)oracle associated with $\mathcal{H}_2$. We transform $\mathcal{E}_1$ to an ITM $\mathcal{E}_2$ that interacts with $\mathcal{O}_{\mathsf{comp}}$ instead of $\mathcal{H}_1$ and $\mathcal{O}_{\mathsf{CCA}}$ instead of $\mathcal{H}_2$, *i.e.* $(\mathcal{E}_2^{\mathcal{O}_{\mathsf{comp}}})^{\mathcal{O}_{\mathsf{CCA}}}$. Since $\mathcal{O}_{\mathsf{CCA}}$ is black-box, this is possible without affecting the output's distribution.

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 $O(k_1)$ 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 4.6. *Let π be a subroutine-respecting k_1-round PPT protocol according to Definition 4.22. Let $\mathcal{O}_{\mathsf{CCA}}$ be a black-box pseudo-oracle and let $\mathcal{O}_{\mathsf{comp}}$ be an oracle capturing $k_2 \in O(1)$ complexity oracles. Let T_1 be the Turing machine comprised of (the honest parties of) π and an adversary $\mathcal{D}_1$ with the interface of the dummy adversary and T_2 be the Turing machine comprised of the environment $\mathcal{Z}$, where T_1 and T_2 are defined as follows.*

- *T_1:*
  - *On input $(1^\kappa, z)$, internally, execute a protocol π and an adversary $\mathcal{D}_1$ interacting with π.*
  - *Receive inputs for parties of π from an external entity and forward outputs of parties of π to an external entity.*
  - *Forward messages from π to the adversary to $\mathcal{D}_1$ and vice versa.*
  - *Forward messages from $\mathcal{D}_1$ intended for the environment to an external identity.*
  - *Forward messages from an external entity marked as coming from $\mathcal{H}$ or from the environment to $\mathcal{D}_1$.*
  - *Messages forwarded from T_1 to an external entity are subject to the (per-entity) bounds of π according to Definition 4.22.*
- *T_2:*
  - *On input $(1^\kappa, z)$, run the environment $\mathcal{Z}$ on input $(1^\kappa, z)$.*
  - *Messages from the environment for $\mathcal{H}$ (and vice versa) are forwarded between $\mathcal{O}_{\mathsf{comp}}$, $\mathcal{O}_{\mathsf{CCA}}$ and the environment.*
  - *Messages from the environment to $\mathcal{H}$ via corrupted parties are forwarded to $\mathcal{O}_{\mathsf{CCA}}$ (and vice versa).*
  - *Messages from $\mathcal{O}_{\mathsf{comp}}$ intended for the adversary to notify that an assumption has been invalidated are forwarded to $\mathcal{O}_{\mathsf{CCA}}$ and an external entity.*
  - *Messages from the environment to the dummy adversary (not intended for $\mathcal{H}$) are forwarded to an external entity.*
  - *Forwarded messages to an external entity are subject to the (per-entity) bounds of π according to $Definition$ 4.22.*
  - *Eventually output what the environment outputs.*

T_1 and T_2 also implement their respective part of the UC control function. Then, the interaction $\langle T_1, T_2^{\mathcal{O}_{\mathsf{comp}}}\rangle^{\mathcal{O}_{\mathsf{CCA}}}(1^\kappa, z)$ has $O(k_1 + k_2)$ rounds according to Definition 4.10.

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

- Messages from $\mathcal{H}$ intended for the adversary to notify that an assumption has been invalidated. The number of such messages is bounded by k_2. Furthermore, by Proposition 4.6, $\mathcal{D}_1$ does not send or receive other messages related to $\mathcal{H}$.

- 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 4.21 and the fact that π is a k_1-round protocol, it is easy to see that the number of these messages is bounded by $O(k_1 + k_2) \in O(k_1)$.

□

We now split up $\mathcal{E}_2$ like in Proposition 4.6. To this end, let $\mathcal{E}_3$ be an ITM that is identical to T_2 that internally emulates the environment $\mathcal{Z}$ and externally interacts with a protocol (π) and an adversary ($\mathcal{D}$).

Formally, this is the execution $\langle \pi \parallel \mathcal{D}, \mathcal{E}_3^{\mathcal{O}_{\mathsf{comp}}}\rangle^{\mathcal{O}_{\mathsf{CCA}}}$ (with omission of the input). We can see the external protocol and dummy adversary resp. simulator for the dummy adversary as machine T_1 and $\mathcal{E}_3$ as T_2 in Proposition 4.6 and conclude that the interaction between external protocol and adversary on the left side and $\mathcal{E}_3$ has $O(k)$ rounds. Also, it is easy to see that $\mathcal{E}_3$ is PPT. By using the black-box property, it thus follows that the statistical distance between the outputs of $\mathcal{E}_2$ and $\mathcal{E}_3$ is negligible.

By using the enhanced $O(k)$-robust composition-order invariance, the enhanced $O(k)$-robust quasi-PPT property (Definitions 4.12 and 4.13) and the black-box property of $\mathcal{O}_{\mathsf{CCA}}$, we can first restrict the access of $\mathcal{O}_{\mathsf{CCA}}$ to $\mathcal{E}_3^{\mathcal{O}_{\mathsf{comp}}}$ and replace $(\mathcal{E}_3^{\mathcal{O}_{\mathsf{comp}}})^{\mathcal{O}_{\mathsf{CCA}}}$ with a PPT ITM $\mathcal{E}_4$ with only access to $\mathcal{O}_{\mathsf{comp}}$, incurring a negligible change in the statistical distance between the outputs only.

Finally, $\mathcal{E}_4$ with access to $\mathcal{O}_{\mathsf{comp}}$ can be replaced by an unbounded TM $\mathcal{E}_5$ without access to $\mathcal{O}_{\mathsf{comp}}$, but with an identical output distribution.

Let $\mathcal{Z}'$ be the UC environment that internally executes $\mathcal{E}_5$ and relays messages between $\mathcal{E}_5$ and the challenge protocol and adversary appropriately. Eventually, $\mathcal{Z}'$ runs the (unbounded) distinguisher D on the output of $\mathcal{E}_5$ and outputs what D outputs. We obtain a distinguishing UC environment $\mathcal{Z}'$ from an Updatable UC environment $\mathcal{Z}$ and distinguisher D, leading to a contradiction of the fact that π statistically UC-emulates ϕ.

We note that in each step, the machine interacting with $\mathcal{O}_{\mathsf{comp}}$ behaves like an admissible adversary. □

Theorem 4.3 can be easily adapted to *e.g.* analogous cases of GUC security. Of course, compatibility is not limited to the cases mentioned in Theorem 4.3 and its variants. However, manual proofs may be necessary.

Meaningfulness. Just like the Imaginary 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 $\mathcal{F}'_{\mathrm{COM}}$ of the ideal functionality for commitments $\mathcal{F}_{\mathrm{COM}}$, which we extend to accept a CRS from the adversary. When the honest committer provides its input v, $\mathcal{F}'_{\mathrm{COM}}$ first checks if the CRS is a valid CRS for the statistically hiding commitment scheme COM_1 of $\mathcal{H}$[12]. Then, it performs the commit phase with the adversary, acting as an honest committer with input v.

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

Thus, Updatable UC security may only guarantee meaningful security for ideal functionalities with less than or equal to k_1 rounds, where $k = k_1 + k_2$, $k_2 \in O(1)$ and if $\mathcal{O}_{\mathsf{CCA}}$ (in $\mathcal{H}$) is black-box, enhanced $O(k)$-robust quasi-PPT and enhanced $O(k)$-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 Updatable UC security. First, we show that Updatable UC security implies UC security for a large class of protocols.

Proposition 4.7 (Justification: UC Security)**.** *Let π, ϕ be PPT protocols such that $\pi \geq_{\text{Upd-UC}} \phi$ (resp. $\pi \geq_{\text{LT-Upd-UC}} \phi$) and i) π and ϕ do not call global functionalities and ii) the simulator never interacts with $\mathcal{H}$ on the committed-value (pseudo-)oracle for the challenge session and iii) $\mathcal{H}$ is black-box. Then, $\pi \geq_{\text{UC}} \phi$ (resp. $\pi \geq_{\text{LT-UC}} \phi$).*

Proof. We prove Proposition 4.7 only for the standard non-long-term notion. The proof for the other case is similar.

Let π, ϕ be protocols such that $\pi \geq_{\text{Upd-UC}} \phi$ and the simulator $\mathcal{S}$ (for the dummy adversary) never queries $\mathcal{H}$ on the committed-value (pseudo-)oracle for the challenge session. Suppose that for the sake of contradiction it holds that $\pi \not\geq_{\text{UC}} \phi$, *i.e.* for every (presumptive) PPT UC simulator $\mathcal{S}'$ 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 $\mathcal{S}'$.

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

We construct an environment $\mathcal{Z}'$ that distinguishes between the Updatable UC execution of π and $\mathcal{D}$ and the Updatable 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.

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 $\mathcal{H}$. When interacting with an environment that never sends `invalidate` messages to $\mathcal{H}$, $\mathcal{S}$ also never queries $\mathcal{H}$ on the complexity part. Thus, in the execution with $\mathcal{Z}'$, $\mathcal{S}$ behaves like a UC simulator and the view of $\mathcal{Z}$ is correctly distributed as in a 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 Updatable UC execution is identical to the distinguishing advantage of $\mathcal{Z}$, leading to a contradiction. $\square$

For the case of ideal functionalities that can be expressed by stand-alone real-ideal security (see *e.g.* [G04; L16]), the following holds regardless of the simulator using the committed-value (pseudo-)oracle of $\mathcal{H}$ or the environment invalidating assumptions.

Proposition 4.8 (Justification: Stand-Alone Security for SFE)**.** *Let $\mathcal{H}$ be a helper with a committed-value (pseudo-)oracle that is black-box and enhanced $O(1)$-robust composition-order invariant and enhanced $O(1)$-robust quasi-PPT. Let π be a N-party PPT protocol in the $\mathcal{F}_{\text{CRS}}$-hybrid model such that π (long-term-) Updatable-UC-realizes $\mathcal{F}_{\text{SFE}}$ (with $\mathcal{H}$) for some function $f : (\{0,1\}^\kappa)^N \times \{0,1\}^{\mathsf{poly}(\kappa)} \to (\{0,1\}^\kappa)^N$. Then, π* securely computes f with abort *in the presence of static malicious adversaries.*

In particular, Proposition 4.8 captures the stand-alone real-ideal security of *e.g.* oblivious transfer and zero-knowledge proof systems. The restriction to protocols in the $\mathcal{F}_{\text{CRS}}$-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 4.8, but note that that the distinguisher in the real-ideal security notion is not provided with a committed-value (pseudo-)oracle (corresponding to an Updatable UC environment that never queries the committed-value (pseudo-)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 [CLP13a] (see also Section 2.5 for a short introduction). Suppose that the committed-value (pseudo-)oracle of $\mathcal{H}$ is black-box, enhanced $O(k)$-robust quasi-PPT and enhanced $O(k)$-robust composition-order invariant and that a PPT protocol π (long-term-) Updatable- or 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 Updatable 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* $\mathcal{H}$).

Proposition 4.9 (Environmental Friendliness of Updatable UC Security). *Let $\mathcal{H}$ be a helper where the committed-value (pseudo-)oracle provided by $\mathcal{H}$ is black-box (Definition 4.9), enhanced $O(k)$-robust composition-order invariant (Definitions 4.11 and 4.13) and enhanced $O(k)$-robust quasi-PPT (Definitions 4.12 and 4.13). Let π be a protocol that Updatable-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 4.9 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 Updatable 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 $\mathcal{O}_{\mathsf{CCA}}$ within $\mathcal{H}$ to replace the simulator $\mathcal{S}$ with access to $\mathcal{H}$ with an efficient simulator[13] $\mathcal{S}'$. 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 4.9 is very similar to the proof in [CLP13b] and the proof of Theorem 4.3 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 security.

Theorem 4.4 (Impossibility of Oblivious Transfer with Long-Term Security). *Let $\mathcal{F}$ be a functionality that is long-term revealing (Definition 2.11) for any party. Then, there is no nontrivial,bilateral[14] PPT protocol π_{OT} that long-term-Rewinding-UC-realizes $\mathcal{F}_{\mathrm{OT}}$ in the $\mathcal{F}$-hybrid model (assuming ideally authenticated communication).*

Theorem 4.4 is a direct consequence of the folklore impossibility result of *correct* statistically secure oblivious transfer in the plain model (even with passive security only). In the following, we give a formal proof, using a similar approach to the one in [MU10]. We note that we can extend Theorem 4.4 to the case of ideally secure communication using a slightly different proof (where the adversary passively corrupts parties to obtain the communication).

[13]For this argument, we only need the committed-value (pseudo-)oracle of $\mathcal{H}$, but not its complexity oracles.

[14]We recall the definition of a bilateral protocol [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."

Proof. For a protocol π_{OT} to long-term-Rewinding-UC-realize $\mathcal{F}_{\mathrm{OT}}$, 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 and also does not invalidate assumptions), (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]:

- $\mathrm{COM}^{m_0,m_1,b}(\dots)$ 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, $\mathrm{COM}^{m_0,m_1,b}(\mathcal{Z}\,\mathsf{S}, \mathsf{S}\,\mathcal{A}, \mathsf{S}\,\mathcal{F})$ 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).

- $\mathrm{OUT}^{m_0,m_1,b}$ denotes the output of the receiver.

- For families of variables $A_{\kappa,z}$ and $B_{\kappa,z}$, we write $A \rhd B$ if there is some probabilistic function G such that $B_{\kappa,z} \overset{s}{\approx} G(\kappa, A_{\kappa,z})$ (and vice versa for $\lhd$). It is easy to see that $\rhd$ and $\lhd$ are transitive. For the sake of an easier notation, we will ignore κ from now on.

We first establish several properties.

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

$$\mathrm{COM}^{m_0,m_1,0}(\mathcal{Z}\,\mathsf{S}, \mathsf{S}\,\mathcal{A}, \mathsf{S}\,\mathcal{F}) \overset{s}{\approx} \mathrm{COM}^{m_0,m_1,1}(\mathcal{Z}\,\mathsf{S}, \mathsf{S}\,\mathcal{A}, \mathsf{S}\,\mathcal{F}) \tag{4.1}$$

Conversely, for long-term sender security, it holds for every m_b, m_{1-b}, m'_{1-b} and $b \in \{0,1\}$ that[15]

$$\mathrm{COM}^{m_0,m_1,b}(\mathcal{Z}\,\mathsf{R}, \mathsf{R}\,\mathcal{A}, \mathsf{R}\,\mathcal{F}) \overset{s}{\approx} \mathrm{COM}^{m_b,m'_{1-b},b}(\mathcal{Z}\,\mathsf{R}, \mathsf{R}\,\mathcal{A}, \mathsf{R}\,\mathcal{F}) \tag{4.2}$$

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

$$m_b \overset{s}{\approx} \mathrm{OUT}^{m_0,m_1,b} \lhd \mathrm{COM}^{m_0,m_1,b}(\mathsf{R}\,\mathcal{A}, \mathsf{R}\,\mathcal{F}) \tag{4.3}$$

$$\mathrm{COM}^{m_0,m_1,b}(\mathsf{S}\,\mathcal{A}, \mathsf{S}\,\mathcal{F}) \rhd m_b \tag{4.4}$$

and, if $m_0 \neq m_1$,

$$m_0 \overset{s}{\approx} \mathrm{OUT}^{m_0,m_1,0} \overset{s}{\not\approx} \mathrm{OUT}^{m_0,m_1,1} \overset{s}{\approx} m_1 \tag{4.5}$$

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

where Equation (4.3) means that the receiver's output (but not necessarily b) can be reconstructed with overwhelming probability from the receiver's communication, Equation (4.4) means that (at least) the result m_b can be reconstructed from the sender's communication (including its communication with $\mathcal{F}$) and Equation (4.5) guarantees that if m_0 and m_1 differ, then for different choice bits, the output of the receiver will be (statistically) different.

Claim 4.5. *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

$$\mathrm{COM}^{m_0,m_1,b}(\mathsf{R}\,\mathcal{A},\mathsf{R}\,\mathcal{F}) \lhd \mathrm{COM}^{m_0,m_1,b}(\mathsf{R}\,\mathcal{A},\mathsf{S}\,\mathcal{F}) \tag{4.6}$$

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

$$\mathrm{COM}^{m_0,m_1,b}(\mathsf{R}\,\mathcal{A},\mathsf{S}\,\mathcal{F}) \lhd \mathrm{COM}^{m_0,m_1,b}(\mathsf{S}\,\mathcal{A},\mathsf{S}\,\mathcal{F}) \tag{4.7}$$

Combining Equations (4.1), (4.3), (4.6) and (4.7), using the definition of $\lhd$ and the transitivity of indistinguishability, we obtain

$$\begin{aligned} m_0 &\overset{s}{\approx} \mathrm{OUT}^{m_0,m_1,0} \overset{s}{\approx} G(\mathrm{COM}^{m_0,m_1,0}(\mathsf{S}\,\mathcal{A},\mathsf{S}\,\mathcal{F})) \\ &\overset{s}{\approx} G(\mathrm{COM}^{m_0,m_1,1}(\mathsf{S}\,\mathcal{A},\mathsf{S}\,\mathcal{F})) \overset{s}{\approx} \mathrm{OUT}^{m_0,m_1,1} \\ &\overset{s}{\approx} m_1 \end{aligned} \tag{4.8}$$

which contradicts Equation (4.5), *i.e.* the correctness, if $m_0 \neq m_1$. □

Claim 4.6. *If $\mathcal{F}$ is long-term revealing for* 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 S and that the communication between S and $\mathcal{A}$ can be computed from the communication between R and $\mathcal{A}$, it follows that for $b \in \{0,1\}$

$$\mathrm{COM}^{m_0,m_1,b}(\mathsf{R}\,\mathcal{A},\mathsf{R}\,\mathcal{F}) \rhd \mathrm{COM}^{m_0,m_1,b}(\mathsf{S}\,\mathcal{A},\mathsf{S}\,\mathcal{F}) \rhd m_b \tag{4.9}$$

Combining Equations (4.1), (4.4) and (4.9) and using the definition of $\rhd$ and the transitivity of indistinguishability, it follows that

$$m_0 \overset{s}{\approx} G(\mathrm{COM}^{m_0,m_1,1}(\mathsf{R}\,\mathcal{A},\mathsf{R}\,\mathcal{F})) \tag{4.10}$$

which contradicts Equation (4.2), *i.e.* the sender security (because m_0 can be computed from R's interaction with $\mathcal{A}$ and $\mathcal{F}$, even though its choice bit was 1).

□

Combining Claims 4.5 and 4.6, the theorem follows. □

Remark 4.24. 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.

4.5. Updatable and Long-Term-Secure Composable Commitment Schemes

When (solely) constructed using (possibly only assumed) hardness assumptions, commitment schemes can be either statistically hiding or statistically binding, but not both. If the hiding property of a commitment scheme requires a hardness assumption, the value committed to may be released as soon as the assumption can be broken. In such a setting, we cannot hope to meaningfully "update" the commitment's security using a new assumption that is believed to be valid.

In contrast, if the commitment scheme is statistically hiding and computationally binding, updates are possible: While the original assumption still holds, one can commit to the same value using a commitment scheme which relies on a different hardness assumption. If the two commitments are consistent, it does not matter if the first one eventually loses its binding property (while the problem for the binding property of the second commitment is still valid).

In the following, we first give an ideal functionality $\mathcal{F}_{\text{UpdCOM}}$ for updatable commitments and a construction that long-term-Updatable-UC-realizes $\mathcal{F}_{\text{UpdCOM}}$ in the $\mathcal{F}_{\text{CRS}}$-hybrid model. Using very similar techniques, we also construct a commitment scheme that long-term-Updatable-UC-realizes $\mathcal{F}_{\text{COM}}$ in the $\mathcal{F}_{\text{CRS}}$-hybrid model for environments that do not invalidate hardness assumptions during the protocol execution.

4.5.1. The Ideal Functionality for Updatable Commitments

The ideal functionality for commitments $\mathcal{F}_{\text{COM}}$ (Figure 2.2) captures a single bit commitment between a committer and a receiver, providing information-theoretic security for both parties. If $\mathcal{F}_{\text{COM}}$ is realized by a protocol that makes use of hardness assumptions, all security may be lost if the used hardness assumption turns out to be wrong.

In the following, we modify $\mathcal{F}_{\text{COM}}$ to capture a setting where the hiding property is guaranteed perfectly, but the (ideal) binding property may be lost if any underlying hardness assumption loses its validity. In such a setting, we want to be able to *update* the commitment by migrating to new assumptions while (some of) the old ones are still valid. This task is captured by the ideal functionality for updatable commitments $\mathcal{F}_{\text{UpdCOM}}$ in Figure 4.2.

Remark 4.25. Being a reactive functionality, $\mathcal{F}_{\text{UpdCOM}}$ may receive inputs when previous phases, *e.g.* updates, have not completed. In particular,

- the commitment may be unveiled while an update is still in progress and

Functionality for Updatable Commitments $\mathcal{F}_{\text{UpdCOM}}$

Parameterized by a security parameter κ, a message space $M = \{0,1\}^\kappa$ and n sets of hardness assumptions $\mathcal{L}_1, \dots, \mathcal{L}_n$. $\mathcal{F}_{\text{UpdCOM}}$ proceeds as follows, interacting with a committer C, a receiver R and an adversary $\mathcal{S}$. Initially, set $\text{cnt} = 1$, $\text{cnt}_{\text{temp}} = 1$ and $\mathit{insecure} = \mathit{false}$.

- Upon receiving $(\texttt{commit}, \mathit{sid}, v)$ from C, where $v \in \{0,1\}^\kappa$:
  - Record the tuple (sid, v) and generate a public delayed output $(\texttt{committed}, \mathit{sid})$ to R. Ignore further `commit` inputs.
- Upon receiving $(\texttt{unveil}, \mathit{sid})$ from C:
  - If there has been no delivered output $(\texttt{committed}, \mathit{sid})$, ignore this input.
  - Generate a public delayed output $(\texttt{unveil}, \mathit{sid}, v)$ to R and halt.
- Upon receiving $(\texttt{unveil}, \mathit{sid}, v')$ with $v' \in \{0,1\}^\kappa$ from C and C is corrupted, proceed as follows:
  - If there has been no delivered output $(\texttt{committed}, \mathit{sid})$, ignore this input.
  - If $\mathit{insecure} = \mathit{false}$, send $(\texttt{query})$ to $\mathcal{H}$ and receive $\mathcal{L}'$. If for every $j \in [\text{cnt}]$ it holds that $\mathcal{L}' \cap \mathcal{L}_j \neq \emptyset$, set $\mathit{insecure} = \mathit{true}$.
  - If $\mathit{insecure} = \mathit{true}$, generate a public delayed output $(\texttt{unveil}, \mathit{sid}, v')$ to R and halt. Otherwise, halt without output.
- Upon receiving $(\texttt{update}, \mathit{sid})$ from party $P \in \{\mathsf{C}, \mathsf{R}\}$:
  - If there has been no delivered output $(\texttt{committed}, \mathit{sid})$, ignore this input.
  - If $\text{cnt}_{\text{temp}} = n$, also ignore the input.
  - Set $\text{cnt}_{\text{temp}} = \text{cnt}_{\text{temp}} + 1$ and generate a public delayed output $(\texttt{updated}, \mathit{sid})$ to R, also informing the adversary of the caller P.
  - When the adversary allows the delivery of the output:
    - If $\mathit{insecure} = \mathit{false}$, send $(\texttt{query})$ to $\mathcal{H}$ and receive $\mathcal{L}'$. If $\mathcal{L}' \cap \mathcal{L}_{\text{cnt}_{\text{temp}}} \neq \emptyset$ or for every $j \in [\text{cnt}]$ it holds that $\mathcal{L}' \cap \mathcal{L}_j \neq \emptyset$, set $\mathit{insecure} = \mathit{true}$.
    - Set $\text{cnt} = \text{cnt} + 1$ and give the output $(\texttt{updated}, \mathit{sid})$ to R.
- Upon receiving $(\texttt{status}, \mathit{sid})$ from C:
  - If there is an outstanding `status` message from the adversary, ignore the input.
  - Send $(\texttt{status}, \mathit{sid})$ to the adversary.
  - Eventually receive an answer $(\texttt{status}, \mathit{sid})$ from the adversary.
  - Generate an output $(\texttt{status}, \mathit{sid}, \text{cnt})$ for C.

Figure 4.2.: Functionality for Updatable Commitments $\mathcal{F}_{\text{UpdCOM}}$.

- the status may be queried while an update is in progress.

Variants of $\mathcal{F}_{\mathrm{UpdCOM}}$ with different behavior are possible.

$\mathcal{F}_{\mathrm{UpdCOM}}$ extends $\mathcal{F}_{\mathrm{COM}}$ in a number of ways. First, $\mathcal{F}_{\mathrm{UpdCOM}}$ is parameterized with sets of hardness assumptions $\mathcal{L}_1, \dots, \mathcal{L}_n$, tracking the assumptions for the binding property of real commitment schemes later to be used to realize $\mathcal{F}_{\mathrm{UpdCOM}}$. Based on the validity of these assumption throughout the execution, $\mathcal{F}_{\mathrm{UpdCOM}}$ determines if the commitment is still considered to be binding.

Initially, the committer commits to a value $v \in \{0,1\}^{\kappa}$.[16] Subsequently, either party may trigger *updates* to “update” the hardness assumption used for the binding property. To this end, $\mathcal{F}_{\mathrm{UpdCOM}}$ keeps a counter cnt which tracks the current set of hardness assumption $\mathcal{L}_{\mathrm{cnt}}$ that is incremented after a successful update phase. Also, a second counter $\mathrm{cnt}_{\mathrm{temp}}$ is used to track *incomplete* update phases. A variable *insecure*, which is initially set to *false*, tracks whether the commitment is still considered to be binding.

Also, an interface for the committer to check an update's status is provided[17]. This is necessary to avoid inconsistencies in protocols where the committer needs to know if the update was successful. For example, the adversary could choose not to deliver the last message of the committer to the receiver. The committer would then consider the update to be completed, while the receiver would not.

An update is considered *successful* if

- the *new* hardness assumption is valid and
- at least one of the *previous* hardness assumptions is still valid when the update has finished.

This is motivated by the intuition that an adversary (in an appropriate real protocol) is unable to *inconsistently* commit if at least one of the previous commitments is still considered to be binding.

In the unveil phase, the receiver learns the original value v, unless the committer is corrupted and *insecure* $=$ *true* because

- one of the updates was unsuccessful or
- for every $i \in [\mathrm{cnt}]$, at least one assumption in $\mathcal{L}_i$ is invalid, where cnt is the counter for the last used hardness assumptions.

In this case, a corrupted committer may change the unveiled value because we consider the commitment to be no longer binding.

[16]Usually, $\mathcal{F}_{\mathrm{COM}}$ allows committing to a bit only. Often, it is useful to support committing to strings instead of obtaining a string commitment scheme via the composition theorem.

[17]In the UC framework, only one message can be sent at the same time. We thus cannot easily make $\mathcal{F}_{\mathrm{UpdCOM}}$ notify both the committer and the receiver of a successful update without the risk of inconsistencies. As the receiver is the party being protected by updating the binding property, we have chosen that it will receive an output by $\mathcal{F}_{\mathrm{UpdCOM}}$ when the update has finished.

Remark 4.26. The criteria for when an update is successful or the commitment is no longer considered to be binding may seem somewhat arbitrary, given that the ideal functionality does not rely on hardness assumptions. However, they are motivated by the properties of two possible realizations:

1. In Section 4.5.2, we realize $\mathcal{F}_{\text{UpdCOM}}$ by repeatedly committing to the value v as well as all previous decommitments $d_1, \ldots, d_j$ in the $j+1$-th commitment (where each commitments is based on a new assumption) using an *extractable* commitment scheme. Eventually, the last commitment is opened, containing v as well as decommitments $d_1, \ldots, d_{\text{cnt}}$. For $j = 1, \ldots, \text{cnt}$, the receiver checks that the j-th commitment is to $(v, d_1, \ldots, d_{j-1})$. If this holds, the receiver accepts v.

   In order to unveil to a different value, one of the following events must occur:

   - There is some index l such that at the end of the $l-1$-th update, the l-th commitment is to a value $v' \neq v$ and valid decommitments $(d_1, \ldots, d_{l-1})$. If this happened when the new l-th as well as any of the previous commitment schemes were (still) considered to be extractable (*i.e.* the update being successful), we could construct a successful adversary against the extractability if one of the commitment schemes with index $< l$ that are still considered to be extractable.
   - There is no such index l such that *the values committed to* lead to a contradiction, but the commitment eventually opened is to a value $v' \neq v$ and valid decommitments $(d_1, \ldots, d_{l-1})$. If at least one commitment scheme were still considered to be extractable, we could reduce to the extractability property.

2. Alternatively, one could update the commitment by repeatedly committing to v and using an argument of knowledge (using the new commitment scheme) to prove that the previous and the new commitment are consistent. Eventually, all commitments are unveiled. The requirements for a cheating committer are similar as in the previous point.

Different protocols for updatable commitment schemes may provide different security guarantees, possibly requiring adjustments to $\mathcal{F}_{\text{UpdCOM}}$.

4.5.2. The Updatable Commitment Scheme π_{UpdCOM}

Having defined the ideal functionality for updatable commitments in Figure 4.2, we continue with a description of the protocol π_{UpdCOM} that long-term-Updatable-UC-realizes $\mathcal{F}_{\text{UpdCOM}}$, *i.e.* guarantees a statistical hiding property and an updatable computational binding property.

π_{UpdCOM} is parameterized with a typed commitment scheme COM (Definition 4.2) that has a non-interactive unveil phase, *i.e.* where the committer only sends a single message to decommit. For the sake of a simpler notation, let COM_i denote the i-th type of COM.

For the initial commitment to v, the first commitment scheme COM_1 is used. Let c_1 denote this first commitment and d_1 the decommitment. For the first update, the commitment scheme COM_2 is used to commit to v and d_1, resulting in decommitment d_2. We refer to this commitment as the *second* commitment. For subsequent updates, the next type of COM is used to commit to v and all previous decommitments $d_1, \ldots, d_{k-1}$. In the unveil phase, the committer sends v along with $d_1, \ldots, d_{\text{cnt}}$. The receiver accepts v if c_j is a commitment to $(v, d_1, \ldots, d_{j-1})$ using decommitment d_j for every $j \in [\text{cnt}]$. We want to show that π_{UpdCOM} long-term-Updatable-UC-realizes $\mathcal{F}_{\text{UpdCOM}}$, *i.e.* that the view of an environment $\mathcal{Z}$ in either execution is statistically indistinguishable from the other.

The intuition behind this protocol's security is as follows: If COM is statistically hiding, so is π_{UpdCOM}.

We can show that if COM is extractable, then a corrupted committer cannot unveil a different value v' unless

- all COM_i are no longer binding (and extractable), presumably allowing the corrupted committer computing different d'_i consistent with v', or
- if one update is performed with a commitment scheme that is not extractable (and binding) during the update, or
- if one update finished "too late", *i.e.* if for the k-th update, *all* $\mathsf{COM}_1, \ldots, \mathsf{COM}_k$ were no longer binding. In this case, the committer can again compute matching decommitments d'_i for $i = 1, \ldots, k$ and commit to $(v', d'_1, \ldots, d_k)$ using the (computationally binding) commitment scheme COM_{k+1}.

Remark 4.27. Before presenting our constructions, we introduce the following conventions. Let COM be a typed commitment scheme, *cf.* Definition 4.2. We import COM into the Updatable UC framework as follows:

- COM induces the description of an ITM for a protocol π_{COM}. As usual, ITIs of instances π_{COM} are associated with a session ID $sid' = (sid||n)$ and a party ID pid, distinguishing between the committer C and the receiver R. The *type* t used for COM is selected by the postfix of the session ID (*i.e.* $sid' = (sid||n)$ selects the n-th type) and we informally denote the scheme as COM_t. (As we do not consider tag-based commitment schemes, the session ID is only used to identify the correct ITI and commitment type, but not used within COM.)
- The CRS of the n-th type is not provided as common input to C and R, but supplied by an instance of $\mathcal{F}_{\text{CRS}}$ with SID $sid' = (sid||n||\texttt{crs})$ parameterized with the Setup algorithm of type n of COM.

By abuse of notation, we subsequently continue to refer to COM instead of π_{COM}.

Construction 5 (Protocol π_{UpdCOM})**.** Parameterized by a security parameter κ and a typed commitment scheme $\mathsf{COM} = (\mathsf{COM}_1, \ldots, \mathsf{COM}_n)$ with non-interactive unveil phase.

Initially, the committer C sets $\text{cnt}_{\mathsf{C}} = 1, \text{continue}_{\mathsf{C}} = \textit{false}$ and the receiver R sets $\text{cnt}_{\mathsf{R}} = 1, \text{continue}_{\mathsf{R}} = \textit{false}$. (Here, cnt_{C} and cnt_{R} are counters for the index of the current type and continue tracks if a commit phase has finished and the protocol may continue with a new phase.)

- On input $(\texttt{commit}, \mathit{sid}, v)$ for C:
  1. C and R execute COM with SID $(\mathit{sid}||1)$ and input v for the committer. Let d_1 denote the decommitment. If the sub-party of the receiver in COM accepts, R outputs $(\texttt{committed}, \mathit{sid})$ and sets $\mathrm{continue}_{\mathsf{R}} = \mathit{true}$. Similarly, C sets $\mathrm{continue}_{\mathsf{C}} = \mathit{true}$. Subsequent `commit` inputs are ignored.
- On input $(\texttt{update}, \mathit{sid})$ for R:
  1. If the commit phase has not finished, ignore the input.
  2. If $\mathrm{cnt}_{\mathsf{R}} = n$ or $\mathrm{continue}_{\mathsf{R}} = \mathit{false}$, ignore the input.
  3. If R has previously sent $(\texttt{update}, \mathit{sid})$ to the committer C and the respective update has not finished, also ignore the input.
  4. Otherwise, send $(\texttt{update}, \mathit{sid})$ to C.
- On input $(\texttt{update}, \mathit{sid})$ for C or message $(\texttt{update}, \mathit{sid})$ from R for C:
  1. If $\mathrm{cnt}_{\mathsf{C}} = n$ or $\mathrm{continue}_{\mathsf{C}} = \mathit{false}$, ignore the input resp. message. Otherwise, set $\mathrm{continue}_{\mathsf{C}} = \mathit{false}$.
  2. C starts an instance of COM with SID $(\mathit{sid}||\mathrm{cnt}_{\mathsf{C}} + 1)$ and input $(v, d_1, \ldots, d_{\mathrm{cnt}_{\mathsf{C}}})$ for the committer.
  3. If $\mathrm{cnt}_{\mathsf{R}} = n$ or $\mathrm{continue}_{\mathsf{R}} = \mathit{false}$, the receiver R ignores the starting commitment. Otherwise, it sets $\mathrm{continue}_{\mathsf{R}} = \mathit{false}$ and participates in the commitment as receiver.
  4. At the end of this commit phase, let $d_{\mathrm{cnt}_{\mathsf{C}}+1}$ denote the decommitment of the finished commitment. C sets $\mathrm{cnt}_{\mathsf{C}} = \mathrm{cnt}_{\mathsf{C}} + 1$ and $\mathrm{continue}_{\mathsf{C}} = \mathit{true}$.
  5. When the sub-party of the receiver of COM with SID $\mathit{sid}||\mathrm{cnt}_{\mathsf{R}} + 1$ accepts, R sets $\mathrm{cnt}_{\mathsf{R}} = \mathrm{cnt}_{\mathsf{R}}+1$, $\mathrm{continue}_{\mathsf{R}} = \mathit{true}$ and outputs $(\texttt{updated}, \mathit{sid})$. Otherwise, R halts.
- On input $(\texttt{status}, \mathit{sid})$ for C:
  1. C sends $(\texttt{status}, \mathit{sid})$ to R.
  2. On receiving $(\texttt{status}, \mathit{sid})$ from C, R sends $(\texttt{status}, \mathit{sid}, \mathrm{cnt}_{\mathsf{R}})$ to C if $\mathrm{continue}_{\mathsf{R}} = \mathit{true}$. Otherwise, the message is ignored.
  3. On receiving $(\texttt{status}, \mathit{sid}, i')$ from R, C outputs $(\texttt{status}, \mathit{sid}, i')$ if $i' = \mathrm{cnt}_{\mathsf{C}}$. Otherwise, it outputs nothing.
- On input $(\texttt{unveil}, \mathit{sid})$ for C:
  1. C sends $(\texttt{unveil}, \mathit{sid}, v, d = (d_1, \ldots, d_{\mathrm{cnt}_{\mathsf{C}}}))$ to R.

2. R outputs $(\texttt{unveil}, sid, v)$ if $|d| = \text{cnt}_\mathsf{R}$ and the j-th commitment opens to $(v, d_1, \ldots, d_{j-1})$ using decommitment d_j for every $j \in [\text{cnt}_\mathsf{R}]$. 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 typed commitment scheme COM and associated assumptions $\mathcal{L}$.

Theorem 4.5. *Let $\mathcal{O}_\mathsf{comp}$ be a deterministic stateless complexity oracle capturing all complexity oracles of the helper $\mathcal{H}$. Let $\mathcal{O}_\mathsf{CCA}$ be a black-box committed-value pseudo-oracle for the typed commitment scheme $\mathsf{COM} = (\mathsf{COM}_1, \ldots, \mathsf{COM}_n)$.*

If COM is an enhanced CCA-binding and enhanced trapdoor typed commitment scheme (Definitions 4.15 and 4.16) with respect to $\mathcal{O}_\mathsf{CCA}$ and $\mathcal{O}_\mathsf{comp}$ and has an appropriate message space, then π_UpdCOM long-term-Updatable-UC-realizes $\mathcal{F}_\text{UpdCOM}$.

Proof. 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 4.4), we consider simulators for the dummy adversary.

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

Definition 4.23 (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.
3. Deliver messages as instructed by the environment to internal simulations of the honest party.
4. Initialize variables $\mathit{insecure} = \mathit{false}$, $\text{cnt} = 1$ and $\mathcal{L} = \emptyset$.
5. When receiving a message $(\texttt{invalidated}, \text{name})$ from $\mathcal{H}$, add name to $\mathcal{L}$.
6. Send $(\texttt{ext-init}, \mathsf{C}, sid, r \xleftarrow{\$} \{0,1\}^\kappa, 1)$ to $\mathcal{H}$ in the name of C and receive $(\texttt{setup}, sid, 1, \text{ck})$ from $\mathcal{H}$. Report ck as output of $\mathcal{F}_\text{CRS}$ with SID $sid||1||\texttt{crs}$. For $j = 2, \ldots, n$, sample $\text{ck}_j \leftarrow \mathsf{Setup}(1^\kappa, j)$. (If the environment has already queried $\mathcal{H}$ on sub-session $(r, 1)$ or later does so, output a special error symbol $\perp$.)[18]

[18]Looking ahead to the proof, we will ignore the negligible error incurred by this abort.

7. Commit phase: Let m denote a commitment message received from the corrupted committer and send $(\texttt{ext-mesg}, \mathsf{C}, \mathit{sid}, r, m)$ to $\mathcal{H}$. If $\mathcal{H}$ answers with $(\texttt{ext-mesg}, \mathsf{C}, \mathit{sid}, r, m')$, report m' as message from R to C. If $\mathcal{H}$ answers with $(\texttt{ext-val}, \mathit{sid}, r, v', \mathit{view})$,
   - output a special symbol $\perp$ if $v' = \perp_{ext}$,
   - halt the simulation of the receiver if $v' = \perp$, *i.e.* the receiver would not accept the commitment,
   - set $v = 0$ if $v' = \texttt{broken}$ and $v = v'$ otherwise.
   - Send $(\texttt{commit}, \mathit{sid}, v)$ to $\mathcal{F}_{\text{UpdCOM}}$ on behalf of C. Also allow the `committed` output of $\mathcal{F}_{\text{UpdCOM}}$ for the receiver.
8. Update phases: Report $\text{ck}_{\text{cnt}+1}$ as output of $\mathcal{F}_{\text{CRS}}$ with SID $\mathit{sid}||\text{cnt}+1||\texttt{crs}$. When the corrupted committer initiates the commit phase of COM with SID $(\mathit{sid}||\text{cnt}+1)$, act like R. When the cnt + 1-th commit phase has finished successfully (from the perspective of R), update *insecure* like $\mathcal{F}_{\text{UpdCOM}}$ would and set $\text{cnt} = \text{cnt} + 1$. If this update has been triggered by an output request $(\texttt{update}, \mathit{sid})$, allow the output. Otherwise, send $(\texttt{update}, \mathit{sid})$ to $\mathcal{F}_{\text{UpdCOM}}$ on behalf of C and subsequently allow the output.
9. Handle `status` messages from the committer like the honest receiver would.
10. Eventually receive a message $(\texttt{unveil}, \mathit{sid}, v', d'_1, \ldots, d'_j)$ from the committer and proceed as follows:
    - If $j \neq \text{cnt}$, halt the simulation of the receiver.
    - If the honest receiver would not accept, halt the simulation of the receiver.
    - If $v' = v$ and the honest receiver would accept, send $(\texttt{unveil}, \mathit{sid})$ to $\mathcal{F}_{\text{UpdCOM}}$ and allow the output.
    - If $v' \neq v$ and the honest receiver would accept, check *insecure* like $\mathcal{F}_{\text{UpdCOM}}$ would.
      - If *insecure* = *true*, send $(\texttt{unveil}, \mathit{sid}, v')$ to $\mathcal{F}_{\text{UpdCOM}}$ and allow the output.
      - If *insecure* = *false*, output a special symbol $\perp$.

In order to prove the validity of the simulator $\mathcal{S}$ in Definition 4.23, we define a number of hybrids. We start with the real execution of π_{UpdCOM} and the dummy adversary $\mathcal{D}$ and gradually change it to an execution of $\mathcal{F}_{\text{UpdCOM}}$ and the simulator $\mathcal{S}$. For each pair of hybrids, we prove the statistical indistinguishability.

- H_0: The real execution with π_{UpdCOM} 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_{UpdCOM} 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 $\mathcal{F}_{\text{UpdCOM}}$ and $\mathcal{S}$.

Claim 4.7. *If $\mathcal{O}_{\mathsf{CCA}}$ 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 $\mathcal{O}_{\mathsf{CCA}}$. □

Claim 4.8. *Let $\mathcal{O}_{\mathsf{comp}}$ be a deterministic stateless complexity oracle capturing all complexity oracles of the helper $\mathcal{H}$. Let $\mathcal{O}_{\mathsf{CCA}}$ be a black-box committed-value pseudo-oracle for* COM. *If* COM *is an enhanced CCA-binding typed commitment scheme (Definition 4.15) with respect to $\mathcal{O}_{\mathsf{CCA}}$ and $\mathcal{O}_{\mathsf{comp}}$, then $out_1 \overset{s}{\approx} out_2$.*

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

We show that $\Pr[\text{E}_\perp] \leq \mathsf{negl}(\kappa)$ for some negligible function negl. To this end, we construct an adversary $\mathcal{B}$ against the enhanced CCA binding property that includes the execution H_1, but plays *all* commitment it receives from the corrupted committer with the experiment. After each commit phase has finished, it receives either the extracted value $e_k \in \{0,1\}^\kappa$ (for the k-th commitment), a special error symbol $\perp_{ext}$ (if the commitment could not be extracted but the commitment scheme is still considered to be extractable), $\perp$ if the receiver did not accept or `broken` if the commitment is considered to be insecure due to invalidated assumptions. We distinguish between the following cases for the extracted value:

- $e_k = \perp$: The receiver would not accept and the execution would not continue. Thus, also halt.
- $e_k =$ `broken`: If the receiver would have accepted in the commit phase (this can be determined by the provided view of the receiver if the commitment scheme is stateless or public-coin), continue the execution. If the receiver would not have accepted (which can be determined similarly), halt.
- $e_k = \perp_{ext}$: Do nothing in this session (yet). If this commitment gets unveiled later on or a valid decommitment is extracted later on, send the decommitment to the game for the associated session, winning it.
- $e_k = (v, d_1, \dots, d_{k-1})$: Perform the following consistency check if $k > 1$: For each previously performed commitment c_i ($i \in [k-1]$), $\mathcal{B}$ checks if $(v, d_1, \dots, d_{i-1})$ can be used to unveil c_i to a different value than the extracted one if the hardness assumptions $\mathcal{L}_i$ associated with c_i are still valid, *i.e.* c_i is still supposed to be extractable. If $\mathcal{B}$ encounters such an inconsistency for a commitment c_i, it sends $((v, d_1, \dots, d_{i-1}), d_i)$ to the game for the session associated with c_i, winning the game.

- e_k is of a different format: Do nothing in this session (yet). If this commitment gets unveiled to a different value later on or a valid decommitment is extracted later on, send the decommitment to the game for the associated session, winning it.

For $\mathrm{E}_\perp$ to occur in H_2, either a) the first commitment is not extractable (despite its assumptions being valid), or b) the corrupted committer must be able to open the last commitment such that the honest receiver would accept a value different from the one committed to in the first commitment, even though the (updatable) commitment is considered to be consistent. In more detail, this means that

- there exists a smallest index i such that e_i (*i.e.* the value committed to in commitment c_i) is inconsistent with e_1 and
- the assumptions for COM_i were valid when the update using COM_i had finished, *i.e.* $\mathcal{L}' \cap \mathcal{L}_i = \emptyset$ and there was a commitment scheme COM_j with $j < i$ such that the update to COM_j was successful and COM_j was still binding after the update to COM_i had finished, *i.e.* $\mathcal{L}' \cap \mathcal{L}_j = \emptyset$.

By definition, $\mathcal{B}$ would win the enhanced CCA-binding game if one of these conditions were met. Thus, $\Pr[\mathrm{E}_\perp]$ can be bounded by the success probability of an adversary in the enhanced CCA-binding game.

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

Let $\mathcal{L}_1, \ldots, \mathcal{L}_n$ denote the sets of hardness assumptions associated with $\mathsf{COM} = (\mathsf{COM}_1, \ldots, \mathsf{COM}_n)$. Define $V(\mathrm{ck}_i, c_i, v_i, d_i) = v_i$ if $\mathsf{OPEN}(\mathrm{ck}_i, c_i, v_i, d_i) = 1$ and $\perp$ otherwise.

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

1. Initially, set $e_1, \ldots, e_n = \perp$, where e_i denotes the (extracted) value committed to in commitment c_i.
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 the corrupted committer with $\mathcal{O}_{\mathsf{CCA}}$. We only sketch how the challenge session is handled. To this end, start a session with type i with $\mathcal{O}_{\mathsf{CCA}}$ for the i-th commitment. Initially, receive a CRS ck_i from $\mathcal{O}_{\mathsf{CCA}}$. Report ck_i as output of $\mathcal{F}_{\mathrm{CRS}}$ with SID $sid||i||\mathtt{crs}$ where sid is the SID of the challenge session in H_1. At the end of the i-th commit phase, set e_i to the extracted value returned by $\mathcal{O}_{\mathsf{CCA}}$ and let c_i denote the corresponding transcript.
3. Simulate $\mathcal{H}$ has follows:
   - Whenever the environment sends $(\mathtt{invalidate}, \mathit{name})$ to $\mathcal{H}$, send $(\mathtt{invalidate}, \mathit{name})$ to the game.
   - Let m be a query for $\mathcal{O}_{\mathit{name}}$. If $\mathcal{Z}$ has previously sent $(\mathtt{invalidate}, \mathit{name})$ to $\mathcal{H}$, send $(\mathtt{oracle}, \mathit{name}, m)$ to $\mathcal{O}_{\mathsf{comp}}$ and forward the answer to $\mathcal{Z}$ as coming from $\mathcal{O}_{\mathit{name}}$. Otherwise, ignore the query.

- Appropriately forward `ext-init` and `ext-mesg` messages between $\mathcal{O}_{\mathsf{CCA}}$ and $\mathcal{Z}$, exposing the same interface to $\mathcal{Z}$ as with $\mathcal{H}$.

4. After each commit phase has finished, do the following, where i is the current index:
   - If $e_i = \bot$, *i.e.* the receiver would not accept, halt.
   - If $e_i =$ `broken`: If the receiver would have accepted in the commit phase (this can be determined by the provided view of the receiver if the commitment scheme is stateless or public-coin), continue the execution. If the receiver would not have accepted (which can be determined similarly), halt.
   - If $e_i = \bot_{ext}$, also continue.
   - If $e_i = (v^i, d_1^i, \ldots, d_{i-1}^i)$ do for $l = 1, \ldots, i-1$: Let $v_l = V(\mathrm{ck}_l, c_l, (v^i, d_1^i, \ldots, d_{l-1}^i), d_l^i)$. If $v_l \neq \bot \wedge v_l \neq e_l$ and no assumption in $\mathcal{L}_l$ has been invalidated, send $((v_j, d_1^i, \ldots, d_{l-1}^i), d_l^i)$ to the enhanced CCA binding game as unveil message for session l.
   - Otherwise, *i.e.* if e_i is of a different format, continue with the execution.
5. If the corrupted committer eventually performs the unveil phase by sending $(v', d_1', \ldots, d_i')$, do the following: For $l = 1, \ldots, i$: Let $v_l = V(\mathrm{ck}_l, c_l, (v', d_1', \ldots, d_{l-1}'), d_l')$ If $v_l \neq \bot \wedge v_l \neq e_l$ and no assumption in $\mathcal{L}_l$ has been invalidated, send $((v', d_1', \ldots, d_{l-1}'), d_l')$ to the enhanced CCA binding game as unveil message for session l.

It is easy to see that $\mathcal{B}$ is an admissible adversary and 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 $\mathcal{O}_{\mathsf{CCA}}$[19], it thus holds that its advantage in the enhanced CCA binding game is greater than or equal to $\Pr[\mathrm{E}_\bot]$. As COM is enhanced CCA-binding by assumption, we can thus bound $\Pr[\mathrm{E}_\bot]$ by a negligible function $\mathsf{negl}_{\mathsf{COM}}^{\text{enh-CCA-binding}}$ for the advantage of an adversary in the enhanced CCA-binding game. The claim follows. $\square$

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

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

Definition 4.24 (Simulator for the Dummy Adversary, Corrupted Receiver)**.** Let $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be the function that, on input (κ, i), returns the length of the decommitment

[19]Formally, we cannot apply the black-box property as-is. This is due to the fact that the number of queries to $\mathcal{O}_{\mathsf{CCA}}$ 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 $\mathcal{O}_{\mathsf{CCA}}$, but the extracted value discarded. Clearly, this does not change the distribution. Coming from this hybrid, we can apply the black-box property.

of the i-th type of COM for a commitment to a message with length κ and security parameter κ for $i = 1$ resp. for a commitment to a message with length $\kappa + f(\kappa, 1) + \cdots + f(\kappa, i-1)$ and security parameter κ for $i > 1$.[20]

- 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, set $\mathrm{cnt} = 1$. For $j = 1, \ldots, n$, sample $(\mathrm{ck}_j, \mathrm{td}_j) \leftarrow \mathsf{TSetup}(1^\kappa, j)$. Report ck_i as output of $\mathcal{F}_{\mathrm{CRS}}$ with SID $sid||i||\mathtt{crs}$.
- When receiving the message $(\mathtt{committed}, sid)$ from $\mathcal{F}_{\mathrm{UpdCOM}}$, use the trapdoor committer algorithm $\mathsf{C}_{\mathrm{trap}}$ of COM on input $(1^\kappa, \mathrm{ck}_1, 1, \kappa, \mathrm{td}_1)$ to perform the commitment with the corrupted receiver. After the commit phase has finished and all messages have been delivered, increment cnt and allow the output.[21]
- When receiving the output notification $(\mathtt{updated}, sid)$ from $\mathcal{F}_{\mathrm{UpdCOM}}$ (as input to C), perform the same checks as the honest committer in π_{UpdCOM} would. If they succeed, perform the $(\mathrm{cnt}+1)$-th commit phase using the trapdoor committer $\mathsf{C}_{\mathrm{trap}}$ of COM on input $(1^\kappa, \mathrm{ck}_{\mathrm{cnt}+1}, \mathrm{cnt}+1, f(\kappa, \mathrm{cnt}), \mathrm{td}_{\mathrm{cnt}+1})$. After the commit phase has finished, increment cnt and allow the output of $\mathcal{F}_{\mathrm{UpdCOM}}$.
- Upon receiving the message $(\mathtt{update}, sid)$ from R, act as in the previous step, except with the following change: After the commit phase has finished, additionally send $(\mathtt{update}, sid)$ to $\mathcal{F}_{\mathrm{UpdCOM}}$ on behalf of R and allow the subsequent output.
- Upon receiving $(\mathtt{status}, sid)$ from $\mathcal{F}_{\mathrm{UpdCOM}}$, perform the same steps as the honest committer. When receiving the $(\mathtt{status})$ message from R, perform the same checks as the honest committer. Upon success, send $(\mathtt{status}, sid)$ to $\mathcal{F}_{\mathrm{UpdCOM}}$, allowing the output.
- When receiving the message $(\mathtt{unveil}, sid, v)$ from $\mathcal{F}_{\mathrm{UpdCOM}}$: For $j = 1, \ldots, \mathrm{cnt}$, use the trapdoor committer $\mathsf{C}_{\mathrm{trap}}$ of COM to create a decommitment d_j for a commitment to $(v, d_1, \ldots, d_{j-1})$ and eventually send $(v, d_1, \ldots, d_{\mathrm{cnt}})$ 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_{UpdCOM} and $\mathcal{D}$.

[20]The existence of such a function f is an additional (but natural) assumption.

[21]As R is corrupted, allowing the output has no visible effect. However, it is necessary for $\mathcal{F}_{\mathrm{UpdCOM}}$ to continue.

- 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_{UpdCOM} 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 creates all but the n-th commitment honestly. For the last commitment, it uses the algorithm of the trapdoor committer of COM like the simulator in Definition 4.24.
- For $i = 1, \ldots, n-1$: H_{2+i}: $\mathcal{F}_{2+i} = \mathcal{F}_2$. $\mathcal{S}_{2+i}$ is defined as $\mathcal{S}_2$, but creates only the first $n-i+1$ commitments honestly. For the remaining commitments, use the algorithm of the trapdoor committer like in Definition 4.24.
- H_{n+2}: The ideal execution with $\mathcal{F}_{\text{UpdCOM}}$ and $\mathcal{S}$.

Claim 4.9. *If $\mathcal{O}_{\mathsf{CCA}}$ 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 4.10. *Let $\mathcal{O}_{\mathsf{comp}}$ be a deterministic stateless complexity oracle capturing all complexity oracles of the helper $\mathcal{H}$. Let $\mathcal{O}_{\mathsf{CCA}}$ be a black-box committed-value pseudo-oracle for* COM. *If* COM *is an enhanced trapdoor commitment scheme with respect to $\mathcal{O}_{\mathsf{CCA}}$ and $\mathcal{O}_{\mathsf{comp}}$, then out_1 and out_2 are statistically indistinguishable.*

Proof. First, we note that out_1 and out_2 are identically distributed if the last update phase (*i.e.* the n-th commitment) is not performed. In the following, we thus consider the case that it is performed.

We prove Claim 4.10 by a reduction to the enhanced trapdoor property of COM (Definition 4.16) with respect to the oracle $\mathcal{O}_{\mathsf{comp}}$ and pseudo-oracle $\mathcal{O}_{\mathsf{CCA}}$.

Let $(\mathcal{B}^{\mathcal{O}_{\mathsf{comp}}})^{\mathcal{O}_{\mathsf{CCA}}}$ be the following PPT adversary against the enhanced trapdoor property (Definition 4.16) of the typed commitment scheme COM:

1. Handle messages for $\mathcal{H}$ like the dummy adversary. In order to emulate $\mathcal{H}$, use the oracles $\mathcal{O}_{\mathsf{comp}}$ and $\mathcal{O}_{\mathsf{CCA}}$ provided by the game.
2. On input $(1^\kappa, z)$, internally start an execution of H_2, but perform the last commitment with the game. More specifically, send $(\texttt{Setup}, n)$ to obtain a setup ck and report ck as output of the instance of $\mathcal{F}_{\text{CRS}}$ with SID $sid||n||\texttt{crs}$.
3. At the beginning of the n-th commit phase, send $(\texttt{Start}, n, (v, d_1, \ldots, d_{n-1}))$ as challenge to the game and forward messages between the game and the corrupted receiver.
4. When receiving the message $(\texttt{unveil}, sid, v)$ from $\mathcal{F}_2$, send $(\texttt{unveil})$ to the game and obtain the decommitment d_n. Send $(v, d_1, \ldots, d_n)$ to the corrupted receiver.

5. Continue the execution.

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

Claim 4.11. *Let $\mathcal{O}_{\mathsf{comp}}$ be a deterministic stateless complexity oracle capturing all complexity oracles of the helper $\mathcal{H}$. Let $\mathcal{O}_{\mathsf{CCA}}$ be a black-box committed-value pseudo-oracle for* COM*. If* COM *is an enhanced trapdoor commitment scheme with respect to $\mathcal{O}_{\mathsf{CCA}}$ and $\mathcal{O}_{\mathsf{comp}}$, then for $i = 1, \dots, n-1$, it holds that $out_{2+i-1} \overset{s}{\approx} out_{2+i}$.*

We omit the proof of Claim 4.11 as it is very similar to the proof of Claim 4.10.

Claim 4.12. *If $\mathcal{O}_{\mathsf{CCA}}$ is black-box, then out_{2+n-1} and out_{n+2} are identically distributed.*

Proof. As the changes between H_{2+n-1} and H_{n+2} are only syntactic and oblivious for the environment, the claim follows from the black-box property of $\mathcal{O}_{\mathsf{CCA}}$. □

As there is a common bound for the distinguishing advantage between adjacent hybrids, it follows that $\mathrm{out}_0 \overset{s}{\approx} \mathrm{out}_{n+2}$ and the claim follows.

Both Parties Honest. This case is very similar to the case of the corrupted receiver and we omit it. □

Remark 4.28. Using a typed commitment scheme COM that is *statistically hiding*, we were able to show that π_{UpdCOM} long-term-Updatable-UC-realizes $\mathcal{F}_{\mathrm{UpdCOM}}$.

Unfortunately, one cannot hope to show that π_{UpdCOM} *computationally* Updatable-UC-realizes $\mathcal{F}_{\mathrm{UpdCOM}}$ (*i.e.* $\pi_{\mathrm{UpdCOM}} \geq_{\mathrm{Upd\text{-}UC}} \mathcal{F}_{\mathrm{UpdCOM}}$) if COM is *computationally* hiding only. If only the receiver is corrupted, the simulator must create the commitment without knowing the honest committer's input. When the hiding property of this commitment is lost (because the underlying hardness assumption gets invalidated), this becomes easily recognizable.

Remark 4.29. Special care has to be taken when replacing $\mathcal{F}_{\mathrm{CRS}}$ by a protocol π. In particular, it is necessary that π long-term- (or even statistically) realizes $\mathcal{F}_{\mathrm{CRS}}$. Otherwise, the commitment scheme may lose its statistical hiding property (even if the distribution of the CRS is computationally indistinguishable).

4.5.3. A Composable Long-Term Secure Commitment Scheme

Using the same building blocks as for Construction 5, we can construct a composable commitment scheme $\pi_{\mathrm{LT\text{-}COM}}$ that is long-term-Updatable-UC-secure for environments that do not invalidate hardness assumptions during the protocol execution.

Let $\pi_{\text{LT-COM}}$ be the protocol that simply executes an appropriate commitment scheme COM (embedded into the execution like denoted in Remark 4.27). Let $\mathcal{S}$ be the simulator that is identical to the simulator for π_{UpdCOM}, except that all update-related messages are ignored. It then follows as a direct corollary from the proof of Theorem 4.5 that the following theorem also holds:

Theorem 4.6. *Let $\mathcal{O}_{\mathsf{CCA}}$ be a black-box committed-value pseudo-oracle for the commitment scheme COM. If COM is a CCA-binding and trapdoor commitment scheme with respect to $\mathcal{O}_{\mathsf{CCA}}$ and has an appropriate message space, then $\pi_{\text{LT-COM}}$ long-term-Rewinding-UC-realizes $\mathcal{F}_{\text{COM}}$.*

Remark 4.30. The property of the environment not sending `invalidate` messages can be relaxed. In particular, Theorem 4.6 also holds for environments that do send `invalidate` messages during the protocol execution, as long as they do not invalidate assumptions necessary for the (enhanced) CCA-binding property of the commitment scheme COM.

Alternatively, one could relax $\mathcal{F}_{\text{COM}}$ similar to $\mathcal{F}_{\text{UpdCOM}}$ to allow a corrupted committer to unveil its commitment to a different value if the assumption(s) for (enhanced) CCA-binding no longer hold.

4.5.4. Possible Instantiations

As stated in Theorem 4.1, the typed PRS commitment scheme (Construction 4) can be instantiated with "base commitment schemes" that are

1. computationally biding and statistically hiding,
2. admit an efficient straight-line trapdoor committer algorithm (in the $\mathcal{F}_{\text{CRS}}$-hybrid model) and
3. have stateless receivers and non-interactive unveil phases.

As we construct our commitment schemes in the $\mathcal{F}_{\text{CRS}}$-hybrid model, there are many such candidate commitment schemes from different assumptions. The stateless requirement is trivial, since our candidate commitment schemes are non-interactive and the receivers are deterministic. Building on Proposition 4.1, we give the following exemplary theorem.

Theorem 4.7. *Let $\mathsf{COM}_{\text{RSA}}$, $\mathsf{COM}_{\text{DLOG}}$ and $\mathsf{COM}_{\text{SIS}}$ be computationally binding and statistically hiding trapdoor commitment schemes with stateless receiver and non-interactive unveil phase based on the RSA assumption resp. the DLOG assumption. resp. the SIS assumption. Let*

- *the RSA assumption hold w.r.t. a complexity oracle for SIS,*
- *the DLOG assumption hold w.r.t. a complexity oracle for RSA and SIS and*
- *the SIS assumption hold w.r.t. a complexity oracle for RSA and DLOG,*

Let COM *be the typed PRS commitment with base commitments* $\mathsf{COM}_{\mathrm{RSA}}$*,* $\mathsf{COM}_{\mathrm{DLOG}}$ *and* $\mathsf{COM}_{\mathrm{SIS}}$*. Let* $\mathcal{L}_1 = \{\mathrm{RSA}, \mathrm{DLOG}\}$*,* $\mathcal{L}_2 = \{\mathrm{DLOG}\}$*,* $\mathcal{L}_3 = \{\mathrm{SIS}\}$*. Then,* π_{UpdCOM} *long-term-Updatable-UC-realizes* $\mathcal{F}_{\mathrm{UpdCOM}}$ *for an appropriate helper* $\mathcal{H}$ *parameterized with* COM*,* $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ *and complexity oracles for RSA, DLOG and SIS.*

A direct consequence of Theorems 4.6 and 4.7 is the following corollary.

Corollary 4.3. *Let* $\mathsf{COM}_{\mathrm{RSA}}$*,* $\mathsf{COM}_{\mathrm{DLOG}}$ *and* $\mathsf{COM}_{\mathrm{SIS}}$ *be computationally binding and statistically hiding trapdoor commitment schemes with stateless receiver and non-interactive unveil phase based on the RSA assumption resp. the DLOG assumption. resp. the SIS assumption. Then, there exist protocols* $\pi^{\mathrm{RSA}}_{\text{LT-COM}}$*,* $\pi^{\mathrm{DLOG}}_{\text{LT-COM}}$*,* $\pi^{\mathrm{SIS}}_{\text{LT-COM}}$ *that long-term-Rewinding-UC-realize* $\mathcal{F}_{\mathrm{COM}}$*.*

4.6. Long-Term-Secure Zero-Knowledge and Commit-and-Prove

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

Theorem 4.8. *If computationally binding, statistically hiding trapdoor commitment schemes with stateless receiver and non-interactive unveil phase exist (in the* $\mathcal{F}_{\mathrm{CRS}}$*-hybrid model), then for every NP relation R, there exists a protocol* π^R_{ZK} *in the* $\mathcal{F}_{\mathrm{CRS}}$*-hybrid model such that* π^R_{ZK} *long-term-Rewinding-UC-realizes* $\mathcal{F}^R_{\mathrm{ZK}}$*.*

The resulting protocol π^R_{ZK} thus features 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 for commit-and-prove (Figure 2.5) for a bounded number of proofs per instance.

Theorem 4.9. *Let* $k \in \mathsf{poly}(\kappa)$*. If computationally binding, statistically hiding trapdoor commitment schemes with stateless receiver and non-interactive unveil phase exist (in the* $\mathcal{F}_{\mathrm{CRS}}$*-hybrid model), then there exists a protocol* π_{CP} *(accepting at most k inputs) in the* $\mathcal{F}_{\mathrm{CRS}}$*-hybrid model such that* π_{CP} *long-term-Rewinding-UC-realizes* $\mathcal{F}_{\mathrm{CP}}$ *(for up to k inputs).*

Being a long-lived functionality, we leave the definition of an updatable variant of $\mathcal{F}_{\mathrm{CP}}$ and its realization in the $\mathcal{F}_{\mathrm{UpdCOM}}$-hybrid model for future work.

In the following, we discuss feasibility results with respect to composable general two-party computation in our framework.

4.7. Long-Term-Secure General Two-Party Computation

Building on the constructions presented in Section 4.6, we now consider protocols for general two-party computation.

We recall our impossibility result (Theorem 4.4), which states that we cannot realize oblivious transfer with long-term security for both parties within our framework from long-term revealing functionalities. In this section, we first consider possible security definitions and then construct a protocol for composable oblivious transfer in the $\mathcal{F}_{\text{CRS}}$-hybrid model where one party is protected with long-term security, which is the best we can hope for in this setting. Using this protocol, we present a protocol for general two-party computation with long-term security for one party.

4.7.1. Defining Security.

A challenge arises when modeling long-term security in a setting where both parties have secret inputs: In order to simulate the party that is only computationally protected, the simulation needs to depend on the party's secrets, as they will eventually leak. Otherwise, the distribution of the environment's view would change between the real and the ideal execution. To this end, a possibility is to adapt the ideal functionality to leak these secrets to the simulator.

In such a setting, trivial (and yet provably secure) protocols realizing such a "leaky" functionality are possible. In these protocols, the party that is supposed to be protected computationally has *no security at all.* To rule out such protocols, we consider protocols that not only long-term-(Rewinding)-UC-realize the (leaky) long-term functionality, but also computationally realize a related functionality without leakage. Additionally, we require that essentially the *same* simulator works in both cases.

In more detail, we consider, without loss of generality, two functionalities for deterministic reactive two-party secure function evaluation (short 2PC SFE), namely a standard UC functionality $\mathcal{F}_{\text{2PC}}$ together with a long-term variant $\mathcal{F}_{\text{lt2PC}}$ (see Figure 4.4) that models the long-term security guarantees and also captures the inevitable (long-term) information leakage.

The functionality $\mathcal{F}_{\text{2PC}}$ is parameterized with functions $f_1, \dots, f_l$ to be evaluated in each round and, in the k-th round, evaluates f_k on the provided inputs as well as the previous round's state. After a round has completed, the adversary may request all outputs (including the outputs of the honest parties). Providing the adversary with the outputs enables simulation in the first place.[22] $\mathcal{F}_{\text{2PC}}$ captures the security properties that can be guaranteed computationally, *i.e.* without the leakage making long-term emulation possible in the first place.

In contrast, $\mathcal{F}_{\text{lt2PC}}$ will also leak, each round, the input of the one party without long-term security, but only after the other party has provided input, guaranteeing the independence of inputs.

In order to see that there are computational security guarantees for parties that are not long-term-protected in the execution with of a protocol π_{lt2PC} that long-term-realizes

[22]We remark that such a functionality can still be used to perform secure function evaluation for secret outputs. Instead of evaluating $f : (x_1, x_2) \mapsto (y_1, y_2)$, the parties evaluate a function $g : ((x_1, s_1), (x_2, s_2)) \mapsto (f(x_1, x_2)_1 \oplus s_1), (f(x_1, x_2)_2 \oplus s_2))$ (where $(f(\cdot, \cdot)_i$ denotes the i-th element of f's output)), *i.e.* mask each party's output using a one-time pad provided with the actual input. Then, the leaked output perfectly hides the "real" output y_i, which can be easily reconstructed.

$\mathcal{F}_{\mathrm{lt2PC}}$, we require that essentially the same simulator is used for both functionalities. To this end, we *trivialize* the simulator that is supposed to interact with $\mathcal{F}_{\mathrm{lt2PC}}$ by providing random input leakage when actually interacting with $\mathcal{F}_{\mathrm{2PC}}$.

Functionality for Reactive Two-Party SFE $\mathcal{F}_{\mathrm{2PC}}$

$\mathcal{F}_{\mathrm{2PC}}$ proceeds as follows, parameterized with a security parameter κ, efficiently computable functions $f_1, \dots, f_l$, where $f_i : (1^\kappa, x_1^i, x_2^i, \mathrm{state}_{i-1}) \mapsto (y_1^i, y_2^i, \mathrm{state}_i)$, running with parties P_1 and P_2 and an adversary $\mathcal{S}$.

1. Initialize a round counter $k = 1$ and set $\mathrm{state}_0 = \perp$.
2. When party P_i gives input $(\texttt{input}, sid, x)$ in the k-th round, store x as x_i^k and send $(\texttt{input-provided}, sid, P_i)$ to the adversary. If x_i^k is already defined, ignore the input.
3. When all parties (both honest and corrupted) have provided input in the k-th round, mark the k-th round as "input finished" and set $(y_1^k, y_2^k, \mathrm{state}_k) = f_k(1^\kappa, x_1^k, x_2^k, \mathrm{state}_{k-1})$.
4. Upon receiving the input $(\texttt{output}, sid)$ from a party P_i, generate a public delayed output $(\texttt{output}, sid, k, y_i^k)$ to P_i.
5. Upon receiving $(\texttt{output}, sid, k)$ from the adversary, send $(\texttt{output}, sid, k, Y_k)$ to the adversary, where Y_k is the set of the outputs in round k. If the k-th round is not marked as "input finished", send $(\texttt{output}, sid, k, \perp)$ to the adversary.
6. When all parties (both honest and corrupted) have received output in the k-th round and $k < l$, set $k = k + 1$.

Figure 4.3.: Ideal Functionality for Reactive Two-Party SFE $\mathcal{F}_{\mathrm{2PC}}$.

We start with a definition of $\mathcal{F}_{\mathrm{2PC}}$ in Figure 4.3 and continue with the definition of an ideal functionality for reactive SFE with long-term security guarantees for one of the two parties in Figure 4.4. To this end, we fix the domain of each party's input and introduce a `lt-query` interface that leaks inputs of the party without long-term security and the outputs of *all* parties. For the party with long-term protection, the leaked inputs are random and independent of the real inputs. To this end, we assume the domains of the parties' inputs and outputs to be efficiently samplable. For the party without long-term protection, the real inputs and outputs are provided.

Remark 4.31. Given that the adversary may learn secrets of an honest party via the `lt-query` interface, $\mathcal{F}_{\mathrm{lt2PC}}$ may not provide sufficient guarantees to be meaningfully used a hybrid functionality.

Functionality for Long-Term Reactive Two-Party SFE $\mathcal{F}_{\text{lt2PC}}^{\mathsf{P}}$

$\mathcal{F}_{\text{lt2PC}}^{msP}$ proceeds as follows, parameterized with a security parameter κ, efficiently computable functions $f_1, \ldots, f_l$, where $f_i : (1^\kappa, x_1^i, x_2^i, \text{state}_{i-1}) \mapsto (y_1^i, y_2^i, \text{state}_i)$, running with parties P_1 and P_2 and an adversary $\mathcal{S}$. Let I_i^j be the domain of P_i's (private) input in the j-th round.

1. Initialize a round counter $k = 1$ and set $\text{state}_0 = \bot$.
2. When party P_i gives input $(\texttt{input}, sid, x)$ in the k-th round, store x as x_i^k and send $(\texttt{input-provided}, sid, P_i)$ to the adversary. If x_i^k is already defined, ignore the input.
3. When all parties (both honest and corrupted) have provided input in the k-th round, mark the k-th round as "input finished" and set $(y_1^k, y_2^k, \text{state}_k) = f_k(1^\kappa, x_1^k, x_2^k, \text{state}_{k-1})$.
4. Upon receiving the input $(\texttt{output}, sid)$ from a party P_i, generate a public delayed output $(\texttt{output}, sid, k, y_i^k)$ to P_i.
5. When all parties (both honest and corrupted) have received output in the k-th round and $k < l$, set $k = k + 1$.
6. Upon receiving the message $(\texttt{lt-query}, sid)$ from the adversary:
   a) Let k' be the index of the last round marked as "input finished".
   b) For $j = 1, \ldots, k'$ and for $i \in \{1, 2\}$:
      - If $\mathsf{P} = P_i$ and P_i is honest, set $x_i'^j \xleftarrow{\$} I_i^j$ and $y_i'^j = y_i^j$.
      - If $\mathsf{P} \neq P_i$ or P_i is corrupted, set $x_i'^j = x_i^j$, $y_i'^j = y_i^j$.
   c) Send $(\texttt{lt-query}, sid, ((x_1'^1, x_2'^1), \ldots, (x_1'^{k'}, x_2'^{k'})), ((y_1'^1, y_2'^1), \ldots, (y_1'^{k'}, y_2'^{k'})))$ to the adversary.

Figure 4.4.: Ideal Functionality for Long-Term Reactive Two-Party SFE $\mathcal{F}_{\text{lt2PC}}^{\mathsf{P}}$.

We now introduce *trivialized* simulators. Let $\mathcal{S}^{\mathsf{P}}$ be a simulator expecting to interact with an instance of $\mathcal{F}_{\text{lt2PC}}$ with long-term security for party P. The trivialized simulator $\mathcal{S}^{\mathsf{P},'}$ for the interaction with $\mathcal{F}_{\text{2PC}}$ internally executes $\mathcal{S}^{\mathsf{P}}$, but returns random inputs for *all* honest parties in (simulated) `lt-query` outputs. Again, the simulator learns the correct outputs, as they are not protected by definition.

Definition 4.25 (Trivialized Simulator)**.** Let $\mathcal{S}^{\mathsf{P}}$ be a simulator expecting to interact with $\mathcal{F}^{\mathsf{P}}_{\mathrm{lt2PC}}$. Let I_i^j be the domain of P_i's (private) input in the j-th round as in $\mathcal{F}^{\mathsf{P}}_{\mathrm{lt2PC}}$. Let $\mathcal{S}^{\mathsf{P},\prime}$ be the *trivialized simulator* for $\mathcal{S}^{\mathsf{P}}$ and $\mathcal{F}^{\mathsf{P}}_{\mathrm{lt2PC}}$ that acts as follows:

- Internally, execute $\mathcal{S}^{\mathsf{P}}$.
- After the i-th round has completed, send $(\texttt{output}, \mathit{sid}, i)$ to $\mathcal{F}_{\mathrm{2PC}}$ to learn that round's outputs.
- Answer $(\texttt{lt-query}, \mathit{sid})$ messages intended for $\mathcal{F}^{\mathsf{P}}_{\mathrm{lt2PC}}$ as follows:
  - For $i \in \{1, 2\}$:
    - Set $x_i'^j \xleftarrow{\$} I_i^j$. If x_i^j has been learned due to party corruption, set $x_i'^j = x_i^j$.
    - y_i^j has been learned as part of the $\texttt{output}$ query to $\mathcal{F}_{\mathrm{2PC}}$ for round j.
  - Return $(\texttt{lt-query}, \mathit{sid}, ((x_1'^1, x_2'^1), \ldots, (x_1'^{k'}, x_2'^{k'})), ((y_1^1, y_2^1), \ldots, (y_1^{k'}, y_2^{k'})))$.
- Forward all other messages intended for $\mathcal{F}^{\mathsf{P}}_{\mathrm{lt2PC}}$ to $\mathcal{F}_{\mathrm{2PC}}$ and report messages coming from $\mathcal{F}_{\mathrm{2PC}}$ as coming from $\mathcal{F}^{\mathsf{P}}_{\mathrm{lt2PC}}$.
- Relay all other messages appropriately.

Remark 4.32. Looking ahead to our constructions, it is important that the simulator may depend on the party P with long-term security: In a two-party protocol in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model, the CRS distribution may change depending on which party has long-term security.

We now define the notion of *non-trivial*[23] *long-term-UC-secure* function evaluation with long-term security for one party. Informally, this notion is satisfied by a protocol $\pi^{\mathsf{P}}_{\mathrm{lt2PC}}$ if $\pi^{\mathsf{P}}_{\mathrm{lt2PC}}$ long-term-UC-realizes $\mathcal{F}^{\mathsf{P}}_{\mathrm{lt2PC}}$ and, for the trivialized simulator, also (non-long-term-) UC-realizes $\mathcal{F}_{\mathrm{2PC}}$. If a protocol π satisfies the proposed notion, then it is easy to see that *all* protocol parties in the long-term execution of π enjoy computational security, even if the simulator always learns the inputs of one party in the ideal execution with $\mathcal{F}^{\mathsf{P}}_{\mathrm{lt2PC}}$.

Definition 4.26. Let $\mathcal{F}_{\mathrm{2PC}}$ be the two-party (reactive) FSE functionality. We say that a protocol π *non-trivially long-term-UC-realizes* $\mathcal{F}_{\mathrm{2PC}}$ *with long-term security for* P if the following holds:

1. $\pi \geq_{\text{LT-UC}} \mathcal{F}^{\mathsf{P}}_{\mathrm{lt2PC}}$, where $\mathcal{F}^{\mathsf{P}}_{\mathrm{lt2PC}}$ is the long-term variant (Figure 4.4) of $\mathcal{F}_{\mathrm{2PC}}$ (Figure 4.3) with long-term security for P. Let $\mathcal{S}^{\mathsf{P}}$ denote the simulator.
2. $\pi \geq_{\text{UC}} \mathcal{F}_{\mathrm{2PC}}$ for the trivialized simulator $\mathcal{S}^{\mathsf{P},\prime}$ of $\mathcal{S}^{\mathsf{P}}$.

[23] Here, non-triviality is related to the question whether the protocol is trivial in the sense that it prematurely discloses honest parties' secrets. It is not to be confused with the notion of non-triviality in Section 3.4.3.

A variant for *non-trivial long-term-Rewinding-UC-secure* function evaluation with long-term Rewinding UC security for one party is defined in total analogy by considering long-term Rewinding UC emulation instead of long-term UC emulation.

For the sake of an easier exposition, we will consider the simpler notion of long-term security of Müller-Quade and Unruh [MU10]. Using Theorem 4.3, long-term Rewinding UC security can be inferred.

Towards realizing $\mathcal{F}_{\text{2PC}}$ and $\mathcal{F}^{\mathsf{P}}_{\text{lt2PC}}$, we now present a protocol for oblivious transfer with long-term security for one party.

4.7.2. Oblivious Transfer with Long-Term Security for One Party

Even given a commitment scheme that is long-term-(Rewinding-)UC-secure, we provably cannot achieve long-term-secure oblivious transfer from long-term revealing setups.

Instead, we construct an OT protocol π_{ltOT} that guarantees long-term security for one party only. To this end, we use a (UC) protocol π_{OT} in the $\mathcal{F}_{\text{CRS}}$-hybrid model that has the following properties, depending on how the CRS is distributed:

- π_{OT} UC-realizes $\mathcal{F}_{\text{OT}}$ and
- in the mode for statistical sender security, denoted by $\pi^{\mathsf{S}}_{\text{OT}}$, it holds that $\pi^{\mathsf{S}}_{\text{OT}}$ statistically UC-realizes $\mathcal{F}_{\text{OT}}$ if the OT receiver is corrupted and the OT sender is honest resp.
- in the mode for statistical receiver security, denoted by $\pi^{\mathsf{R}}_{\text{OT}}$, it holds that $\pi^{\mathsf{R}}_{\text{OT}}$ statistically UC-realizes $\mathcal{F}_{\text{OT}}$ if the OT sender is corrupted and the OT receiver is honest and
- it is possible to prove the correctness of the execution relative to committed inputs.[24]

The dual-mode construction of Peikert, Vaikuntanathan, and Waters [PVW08] satisfies these conditions. In particular, it can be switched between statistical security for the sender resp. the receiver and is in the $\mathcal{F}_{\text{CRS}}$-hybrid model. If the helper under consideration has appropriate properties (see Theorem 4.3), we can import π_{OT} into our framework, achieving Rewinding UC security resp. long-term Rewinding UC security.

Remark 4.33. The protocol of Peikert, Vaikuntanathan, and Waters [PVW08] achieves statistical UC security *only* when either the receiver is corrupted (and the sender is honest and enjoys statistical security) or when the sender is (passively) corrupted (and the receiver honest and enjoys statistical security): Due to the corruption, the simulator learns the secrets of the party without statistical security and can perform a statistically correct simulation. If the party without statistical security were not corrupted, this would be impossible. Conversely, simulation in [PVW08] is not possible when the party with statistical security is corrupted. (Of course, computational security with arbitrary

[24]This may not be possible depending on the used setup, *e.g.* in the case of random oracles. In the following, we consider protocols in the $\mathcal{F}_{\text{CRS}}$-hybrid model where this is possible.

corruptions is achieved by [PVW08].) We will enable extraction of all corrupted parties through the use of $\mathcal{F}_{\mathrm{CP}}$, which we can later on replace with a long-term-secure protocol in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model.

The protocol π_{ltOT} works as follows. First, each party uses an instance of the commit-and-prove functionality $\mathcal{F}_{\mathrm{CP}}$ to commit to its input and its randomness. Subsequently, both parties execute π_{OT} 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 $\mathcal{F}_{\mathrm{CP}}$ allows for easy extraction resp. equivocation.

We use the definitional approach outlined in Section 4.7.1. To this end, we consider the ideal functionality for SFE $\mathcal{F}_{\mathrm{2PC}}$ with the following parameters:

- $I^1_{\mathsf{R}} = \{0,1\}$, $I^1_{\mathsf{S}} = \{0,1\}^\kappa \times \{0,1\}^\kappa$,
- $f_1 : (1^\kappa, (m_0, m_1), b, \bot) \mapsto (\bot, m_b, \bot)$.

In the following, we consider the long-term variants $\mathcal{F}^{\mathsf{R}}_{\mathrm{lt2PC}}$ and $\mathcal{F}^{\mathsf{S}}_{\mathrm{lt2PC}}$, where the first long-term-protects the receiver and the second long-term-protects the sender. Note that the other party is not long-term-protected, even when both parties are honest. In the following, we will refer to these long-term variants of OT by $\mathcal{F}^{\mathsf{R}}_{\mathrm{ltOT}}$ and $\mathcal{F}^{\mathsf{S}}_{\mathrm{ltOT}}$.

Construction 6 (The OT Protocol π'_{ltOT})**.**
Parameterized with an OT protocol π_{OT}.

1. On input $(\texttt{input}, \mathit{sid}, b)$, the OT receiver R samples randomness r_{R} and sends $(\texttt{commit}, \mathit{sid}||\mathsf{R}, (b, r_{\mathsf{R}}))$ to an instance of $\mathcal{F}_{\mathrm{CP}}$ with SID $\mathit{sid}||\mathsf{R}$.

2. On input $(\texttt{input}, \mathit{sid}, (m_0, m_1))$, the OT sender S samples randomness r_{S} and sends $(\texttt{commit}, \mathit{sid}||\mathsf{S}, (m_0, m_1, r_{\mathsf{S}}))$ to an instance of $\mathcal{F}_{\mathrm{CP}}$ with SID $\mathit{sid}||\mathsf{S}$.

3. R and S keep a list of received messages $\overline{m}_{\mathsf{R}}$ resp. $\overline{m}_{\mathsf{S}}$.

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 $(\texttt{CP-prover}, \mathit{sid}||\mathsf{S}, (m, \overline{m}_{\mathsf{S}}))$ to $\mathcal{F}_{\mathrm{CP}}$ with the relation $R^{\pi_{\mathrm{OT}}}_{\mathsf{S}} = \{(m, \overline{m}_{\mathsf{S}}), (m_0, m_1, r_{\mathsf{S}}) \mid m = \pi_{\mathrm{OT}}((m_0, m_1), \overline{m}_{\mathsf{S}}; r_{\mathsf{S}})\}$ 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 $(\texttt{CP-proof}, \mathit{sid}||\mathsf{S}, (m, \overline{m}_{\mathsf{S}}))$ from $\mathcal{F}_{\mathrm{CP}}$.

6. Similarly, for every message m sent to S by R, R sends $(\texttt{CP-prover}, \mathit{sid}||\mathsf{R}, (m, \overline{m}_{\mathsf{R}}))$ to $\mathcal{F}_{\mathrm{CP}}$ with the relation $R^{\pi_{\mathrm{OT}}}_{\mathsf{R}} = \{(m, \overline{m}_{\mathsf{R}}), (b, r_{\mathsf{R}}) \mid m = \pi_{\mathrm{OT}}(b, \overline{m}_{\mathsf{R}}; r_{\mathsf{R}})\}$ 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 $(\texttt{CP-proof}, \mathit{sid}||\mathsf{R}, (m, \overline{m}_{\mathsf{R}}))$ from $\mathcal{F}_{\mathrm{CP}}$.

7. When the receiver of π_{OT} outputs m_b, R stores m_b.

8. On input $(\texttt{output}, \mathit{sid})$, R outputs $(\texttt{output}, \mathit{sid}, 1, m_b)$ and S outputs $(\texttt{output}, \mathit{sid}, 1, \perp)$.

Theorem 4.10. *Let π_{OT} be a protocol in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model such that*

- $\pi_{\mathrm{OT}} \geq_{\mathrm{UC}} \mathrm{IDEAL}(\mathcal{F}_{\mathrm{OT}})$ *and*
- $\pi_{\mathrm{OT}}^{\mathsf{S}} \geq_{\text{Stat-UC}} \mathrm{IDEAL}(\mathcal{F}_{\mathrm{OT}})$ *if the sender is honest and the receiver is corrupted resp.*
- $\pi_{\mathrm{OT}}^{\mathsf{R}} \geq_{\text{Stat-UC}} \mathrm{IDEAL}(\mathcal{F}_{\mathrm{OT}})$ *if the receiver is honest and the sender is corrupted,*

where $\pi_{\mathrm{OT}}^{\mathsf{S}}$ is π_{OT} in the mode for statistical sender security and $\pi_{\mathrm{OT}}^{\mathsf{R}}$ is π_{OT} in the mode for statistical receiver security. Then, π'_{ltOT} is a protocol in the $(\mathcal{F}_{\mathrm{CRS}}, \mathcal{F}_{\mathrm{CP}})$-hybrid model such that

- $\pi_{\mathrm{ltOT}}^{\prime,\mathsf{S}} \geq_{\text{Stat-UC}} \mathrm{IDEAL}(\mathcal{F}_{\mathrm{ltOT}}^{\mathsf{S}})$ *and* $\pi_{\mathrm{ltOT}}^{\prime,\mathsf{S}} \geq_{\mathrm{UC}} \mathrm{IDEAL}(\mathcal{F}_{\mathrm{OT}})$ *for the trivialized simulator resp.*
- $\pi_{\mathrm{ltOT}}^{\prime,\mathsf{R}} \geq_{\text{Stat-UC}} \mathrm{IDEAL}(\mathcal{F}_{\mathrm{ltOT}}^{\mathsf{R}})$ *and* $\pi_{\mathrm{ltOT}}^{\prime,\mathsf{R}} \geq_{\mathrm{UC}} \mathrm{IDEAL}(\mathcal{F}_{\mathrm{OT}})$ *for the trivialized simulator.*

Here, $\pi_{\mathrm{ltOT}}^{\prime,\mathsf{P}}$ uses $\pi_{\mathrm{OT}}^{\mathsf{P}}$ as sub-protocol.

Looking ahead, we will obtain a protocol π_{ltOT} with long-term Rewinding UC security by plugging in the construction for $\mathcal{F}_{\mathrm{CP}}$ in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model from Theorem 4.9 into π'_{ltOT}.

First, we provide a sketch of the proof of Theorem 4.10.

Proof sketch. We state the necessary simulators and give an intuition why the simulation is correct.

Definition 4.27 (Simulator for the Dummy Adversary, Corrupted Sender)**.**

1. Simulate the instance of $\mathcal{F}_{\mathrm{CRS}}$ which is part of π_{OT} by returning a CRS that is distributed as the CRS in π_{OT}.

2. Appropriately simulate both instances of $\mathcal{F}_{\mathrm{CP}}$.

3. When receiving $(\texttt{input-provided}, \mathit{sid}, \mathsf{R})$ from $\mathcal{F}_{\mathrm{ltOT}}$, report $(\texttt{receipt}, \mathit{sid}||\mathsf{R})$ as output from $\mathcal{F}_{\mathrm{CP}}$.

4. Read the first input to $\mathcal{F}_{\mathrm{CP}}$ with SID $\mathit{sid}||\mathsf{S}$ to obtain m_0, m_1 and send m_0, m_1 to $\mathcal{F}_{\mathrm{ltOT}}$ as input of the corrupted sender.

5. When receiving the output request for R, send $\texttt{lt-query}$ to $\mathcal{F}_{\mathrm{ltOT}}$ and receive $(\texttt{lt-query}, \mathit{sid}, ((m_0, m_1), b), (\perp, m_b))$.

6. Execute π'_{ltOT} as the honest receiver would, using input b obtained from the `lt-query` answer.

7. When the internally emulated honest receiver has finished, allow the output of $\mathcal{F}_{\mathrm{ltOT}}$ for R.

The simulator for the corrupted sender learns the correct input of the corrupted sender through $\mathcal{F}_{\mathrm{CP}}$. If $\mathcal{F}_{\mathrm{ltOT}}$ models long-term sender security, the simulator learns the correct input of R. Otherwise, the leaked input of R is random. Thus, if π_{OT} provides statistical sender security, the correct input for R is used. If π_{OT} provides statistical receiver security, it does not matter if b is correct, as it is statistically protected (and the correct (m_0, m_1) are used).

Definition 4.28 (Simulator for the Dummy Adversary, Corrupted Receiver)**.**

1. Simulate the instance of $\mathcal{F}_{\mathrm{CRS}}$ which is part of π_{OT} by returning a CRS that is distributed as the CRS in π_{OT}.

2. Appropriately simulate both instances of $\mathcal{F}_{\mathrm{CP}}$.

3. When receiving $(\texttt{input-provided}, \mathit{sid}, \mathsf{S})$ from $\mathcal{F}_{\mathrm{ltOT}}$, report $(\texttt{receipt}, \mathit{sid}||\mathsf{S})$ as output from $\mathcal{F}_{\mathrm{CP}}$.

4. Read the first input to $\mathcal{F}_{\mathrm{CP}}$ with SID $\mathit{sid}||\mathsf{R}$ to obtain b and send b to $\mathcal{F}_{\mathrm{ltOT}}$ as input of the corrupted receiver.

5. When the sender has provided its input, send `lt-query` to $\mathcal{F}_{\mathrm{ltOT}}$ and receive $(\texttt{lt-query}, \mathit{sid}, ((m_0, m_1), b), (\perp, m_b))$.

6. Execute π_{ltOT} as the honest sender would, using
   - m_0 from the receiver output m_b and m_1 from the sender input in the `lt-query` answer if the extracted b is 0,
   - m_1 from the receiver output m_b and m_0 from the sender input in the `lt-query` answer if the extracted b is 1.

7. When the internally emulated honest sender finished, allow the output of $\mathcal{F}_{\mathrm{ltOT}}$ for S.[25]

The simulator for the corrupted receiver learns the correct input of the corrupted receiver through $\mathcal{F}_{\mathrm{CP}}$. If $\mathcal{F}_{\mathrm{ltOT}}$ models long-term sender security, then the reported sender input (m_0, m_1) is random. If $\mathcal{F}_{\mathrm{ltOT}}$ models long-term receiver security, then the reported sender input (m_0, m_1) is correct. In any case, the receiver's output m_b is correct. Thus, the simulator always uses m_b from the receiver's output and m_{1-b} from the sender input. If π_{OT} provides statistical sender security, m_{1-b} is statistically protected and it does not matter whether this value is correct or not. If π_{OT} provides statistical receiver security, m_{1-b} leaks but is correct. Conversely, b and m_b are always correct.

[25]This output is $\perp$, but per definition of $\mathcal{F}_{\mathrm{2PC}}/\mathcal{F}_{\mathrm{lt2PC}}$, all parties get output.

Definition 4.29 (Simulator, Both Parties Honest)**.**

1. Simulate the instance of $\mathcal{F}_{\mathrm{CRS}}$ which is part of π_{OT} by returning a CRS that is distributed as the CRS in π_{OT}.
2. Appropriately simulate both instances of $\mathcal{F}_{\mathrm{CP}}$.
3. When receiving $(\texttt{input-provided}, \mathit{sid}, \mathsf{S})$ resp. $(\texttt{input-provided}, \mathit{sid}, \mathsf{R})$, report $(\texttt{receipt}, \mathit{sid}||\mathsf{S})$ resp. $(\texttt{receipt}, \mathit{sid}||\mathsf{R})$ as output from $\mathcal{F}_{\mathrm{CP}}$.
4. When both parties have provided input, send `lt-query` to $\mathcal{F}_{\mathrm{ltOT}}$ and receive $(\texttt{lt-query}, \mathit{sid}, ((m_0, m_1), b), (\perp, m_b))$.
5. Honestly execute the protocol using the following inputs:
   - Use b from the receiver's input
   - Use m_b from the receiver's output.
   - Use m_{1-b} from the sender's input, where b is from the receiver's input.
6. When the protocol execution has finished, allow the outputs of $\mathcal{F}_{\mathrm{ltOT}}$.

The simulator for the case that both parties are honest can only rely on inputs and outputs learned from $\mathcal{F}_{\mathrm{ltOT}}$. If $\mathcal{F}_{\mathrm{ltOT}}$ models long-term receiver security, then the reported sender input (m_0, m_1) is correct (and so is m_b). Thus, the correct input for the sender is used. Conversely, when $\mathcal{F}_{\mathrm{ltOT}}$ models long-term sender security, then the correct b and m_b are used.

We now consider UC security with the trivialized simulator. To this end, we define the following hybrids for $\mathsf{P} \in \{\mathsf{S}, \mathsf{R}\}$:

- H_0: The real UC execution with $\pi_{\mathrm{ltOT}}^{\prime,\mathsf{P}}$ and the dummy adversary $\mathcal{D}$.
- H_1: The ideal UC execution with $\mathcal{F}_{\mathrm{OT}}$ and a simulator $\mathcal{S}_1$ that is defined as follows: Execute the UC simulator $\mathcal{S}$ for π_{OT}, but additionally simulate $\mathcal{F}_{\mathrm{CP}}$. In particular, simulate the outputs of $\mathcal{F}_{\mathrm{CP}}$ for simulated messages of honest protocol parties.
- H_2: The ideal UC execution with $\mathcal{F}_{\mathrm{OT}}$ and the simulator $\mathcal{S}_2 = \mathcal{S}_1$. The honest parties' inputs to $\mathcal{F}_{\mathrm{OT}}$ are changed as the trivialized simulator would report them as the result of a `lt-query` message. The honest parties' outputs are generated relative to their real (unchanged) inputs.
- H_3: The ideal UC execution with $\mathcal{F}_{\mathrm{OT}}$ and the trivialized simulator $\mathcal{S}^{\mathsf{P},\prime}$.

It is easy to see that the outputs of the environment in H_0 and H_1 are computationally indistinguishable due to the UC security of π_{OT}.

As the environment's view is identically distributed in H_1 and H_2 by definition, its outputs in both hybrids are identically distributed.

In H_3, the execution is distributed like a *real* execution with the same inputs and outputs as in H_2 due to the definition of the trivialized simulator. As π_{OT} UC-realizes $\mathcal{F}_{\mathrm{OT}}$, the (computational) indistinguishability of the environment's output follows.

The case of statistical UC security is trivial and we omit it.

□

We note that π'_{ltOT} is constant-round if π_{OT} is constant-round, a property fulfilled by the construction in [PVW08]. If the committed-value oracle of $\mathcal{H}$ has appropriate properties, π'_{ltOT} fulfills long-term Rewinding UC security and we can use the composition theorem to replace $\mathcal{F}_{\mathrm{CP}}$ with an appropriate protocol that long-term-Rewinding-UC-realizes $\mathcal{F}_{\mathrm{CP}}$. The resulting protocol π_{ltOT} then non-trivially realizes $\mathcal{F}_{\mathrm{OT}}$ with long-term Rewinding UC security.

4.7.3. Two-Party Secure Function Evaluation with Long-Term Security for One Party

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

As we are interested in protocols with as few rounds as possible, we explore an approach based on garbled circuits [Y86]. First, we note that most garbling schemes only provide computational security. While constructions with information-theoretic security exist (*e.g.* [K05]), 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 garbling scheme, interestingly, does not need to provide information-theoretic security. Indeed, if the party P with information-theoretic security is known in advance (as we assume for the OT protocol), we let the other party perform the garbling. If the OT protocol provides long-term security for P, then there is no information flow depending on P's input to the other party, except when it learns the result.

Non-Reactive Two-Party Computation. We start with a sketch of the construction π_{nr2PC} for *non-reactive* two-party computation. π_{nr2PC} uses a projective garbling scheme (see [BHR12] for an introduction and definitions) with privacy as well as an OT protocol with long-term receiver security in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model as building blocks.

First, P_1 and P_2 commit to their input and randomness using $\mathcal{F}_{\mathrm{CP}}$ like in Construction 6. Without loss of generality, we assume that P_1 is the party with long-term security. Let $f : (1^\kappa, x_1, x_2) \mapsto (y_1, y_2)$ be the function P_1 and P_2 jointly want to evaluate.

The general idea is as follows: P_2 prepares a garbled circuit for f as well as input labels. It sends the garbled circuit to P_1, along with the labels for its input x_2. Also, P_2 proves via $\mathcal{F}_{\mathrm{CP}}$ that the garbling was performed correctly relative to (x_2, r_2), where x_2 is P_2's actual input and r_2 is a random string that is used as random tape in the subsequent execution. Then, P_1 and P_2 use instances of $\pi^{\mathsf{R}}_{\mathrm{OT}}$ that provide long-term receiver security. In each instance, (a sub-party of) P_2 acts as OT sender, using the input labels for P_1's input as the OT sender's input. Conversely, (a sub-party of) P_1

uses the corresponding bit of x_1 as choice bit. For each step, like in Construction 6, both parties prove the correctness using $\mathcal{F}_{\mathrm{CP}}$ relative to their corresponding inputs and randomness. Also, P_2 sends the output labels to P_1, allowing it to later reconstruct the output. Again, correctness is proven via $\mathcal{F}_{\mathrm{CP}}$. P_1 can now evaluate the circuit, learning its own output y_1 and the output y_2 of P_2. It sends the output y_2 to P_2, again proving via $\mathcal{F}_{\mathrm{CP}}$ that y_2 was obtained correctly. With the exception of the final message for P_2's output, there is only a (long-term) information flow from P_2 to P_1 in the above protocol, but not vice versa.

Let $\mathcal{F}_{\mathrm{ltnr2PC}}$ resp. $\mathcal{F}_{\mathrm{nr2PC}}$ denote the non-reactive variants of $\mathcal{F}_{\mathrm{lt2PC}}$ resp. $\mathcal{F}_{\mathrm{2PC}}$. It is easy to prove that this protocol long-term-UC-realizes $\mathcal{F}_{\mathrm{ltnr2PC}}$: If P_1 is honest and P_2 is corrupted, the simulator learns P_1's input (which is not long-term-protected) via $\mathcal{F}_{\mathrm{ltnr2PC}}$ and can simulate P_1 by essentially executing the real protocol of P_1. P_2's input is learned via its input to $\mathcal{F}_{\mathrm{CP}}$. If P_1 is corrupted, the simulator can extract its input via its input to $\mathcal{F}_{\mathrm{CP}}$. In order to simulate the message for P_2's output, it can simply use the output y_2 learned from $\mathcal{F}_{\mathrm{ltnr2PC}}$ and simulate the message from $\mathcal{F}_{\mathrm{CP}}$ that y_2 is correct. If both parties are honest, the simulator learns P_2's input x_2 from $\mathcal{F}_{\mathrm{ltnr2PC}}$ and can construct a correctly distributed circuit with corresponding labels. For the final message for P_2's output, it uses y_2 learned from $\mathcal{F}_{\mathrm{ltnr2PC}}$.

Unfortunately, it is not possible to prove that this protocol (non-long-term-)UC-realizes $\mathcal{F}_{\mathrm{nr2PC}}$ for essentially the same simulator as in the long-term case: In the above sketch, the simulator garbles the circuit dependent on the garbler's input, which it does not know when interacting with $\mathcal{F}_{\mathrm{nr2PC}}$. In principle, a simulation using the simulator for the garbled circuit scheme (which does not depend on the garbler's input) is possible—however, this would lead to a different simulator not fulfilling the requirements of Definition 4.26.

To address this problem, we evaluate a function g instead of f, where g performs an additional masking step of the outputs, always leading to results that looks uniformly random for the evaluator, *i.e.* P_1. In more detail, g takes input x_1 from P_1 and (x_2, s_1, s_2) from P_2 and returns $y_1' = y_1 \oplus s_1$ as output for P_1 and $y_2' = y_2 \oplus s_2$ as output for P_2, where y_1 and y_2 are the outputs of f for P_1 and P_2 on inputs x_1 and x_2. Finally, P_1 sends y_2' to P_2 and P_2 sends s_1 to P_1, allowing each party to obtain and unmask the result, which they can then return as output. Additionally, the parties prove via $\mathcal{F}_{\mathrm{CP}}$ that y_2' resp. s_1 are correct in the sense that they correspond to the garbled circuit's output resp. input. The simulator, when learning the result from the ideal functionality, can send simply compute and send an appropriate s_2 to equivocate P_1's result resp. an appropriate y_2'.

For the sake of an easier presentation, we assume that the inputs x_1 and x_2 are bitstrings of length κ.

Construction 7 (The Protocol π_{nr2PC})**.** Let $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ be a garbling scheme according to [BHR12]. Let $f : (x_1, x_2) \mapsto (y_1, y_2)$ be the function P_1 and P_2 want to evaluate.

1. On input $(\mathtt{input}, sid, x_1)$ for party P_1, P_1 samples $r_1 \xleftarrow{\$} \{0,1\}^{\mathsf{poly}(\kappa)}$ and sends $(\mathtt{commit}, sid||P_1, x_1, r_1)$ to an instance of $\mathcal{F}_{\mathrm{CP}}$ with SID $sid||P_1$.

2. On input $(\texttt{input}, \mathit{sid}, x_2)$ for party P_2, P_2 waits for $(\texttt{receipt}, \mathit{sid}||P_1)$ from an instance of $\mathcal{F}_{\mathrm{CP}}$ with SID $\mathit{sid}||P_1$ and then samples $s_1, s_2, r_2 \xleftarrow{\$} \{0,1\}^{\mathsf{poly}(\kappa)}$ and sends $(\texttt{commit}, \mathit{sid}||P_2, x_2, s_1, s_2, r_2)$ to an instance of $\mathcal{F}_{\mathrm{CP}}$ with SID $\mathit{sid}||P_2$.

3. Let $f(x_1, x_2)_i = y_i$, *i.e.* the output of f for party P on input (x_1, x_2). Define $g(x_1, (x_2, s_1, s_2)) := (f(x_1, x_2)_1 \oplus s_1, f(x_1, x_2)_2 \oplus s_2)$.

4. After having received $(\texttt{receipt}, \mathit{sid}||P_1)$ from $\mathcal{F}_{\mathrm{CP}}$ with SID $\mathit{sid}||P_1$, P_2 computes $(F, e = X_1^{1,0}, X_1^{1,1}, \ldots, X_1^{\kappa,1}, X_2^{1,0}, \ldots, X_2^{l,1}, \ldots, d) \leftarrow \mathsf{Gb}(1^\kappa, g)$, using a fresh part of r_2 as randomness. Here, l denotes the length of x_2, s_1 and s_2.

5. P_2 sends $(F, (X_2^1 = X_2^{1,x_2'[1]}, \ldots, X_2^\kappa = X_2^{l,x_2'[l]})), d)$ to P_1, proving correctness via $\mathcal{F}_{\mathrm{CP}}$. Here, x_2' denotes (x_2, s_1, s_2).

6. P_1 and P_2 start κ sessions of $\pi_{\mathrm{OT}}^{\mathsf{R}}$ in parallel with P_1 acting as receiver and P_2 as sender. In the i-th session, P_1 uses $x_1[i] \in \{0,1\}$ as input while P_2 uses $(X_1^{i,0}, X_1^{i,1})$ as input. Let $(X_1^1, \ldots, X_1^\kappa)$ denote the result for P_1. As in Construction 6, both parties prove correctness via $\mathcal{F}_{\mathrm{CP}}$ relative to their initial input.[26]

7. P_1 computes $Y = \mathsf{Ev}(F, (X_1^1, \ldots, X_1^\kappa, X_2^1, \ldots, X_2^l))$ and $(y_1', y_2') = \mathsf{De}(Y, d)$.

8. P_1 sends y_2' to P_2 and proves correctness via $\mathcal{F}_{\mathrm{CP}}$.

9. Upon receiving y_2' and the proof of correctness via $\mathcal{F}_{\mathrm{CP}}$, P_2 stores $y_2 = y_2' \oplus s_2$ as result and sends s_1 to P_1.

10. Upon receiving s_1 from P_2, P_1 stores $y_1 = y_1' \oplus s_1$ as result.

11. On input $(\texttt{output}, \mathit{sid})$, P_i outputs $(\texttt{output}, \mathit{sid}, y_i)$ if y_i has been previously stored.

Proposition 4.10. *Let π_{OT} be a protocol in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model such that*

- $\pi_{\mathrm{OT}} \geq_{\mathrm{UC}} \mathrm{IDEAL}(\mathcal{F}_{\mathrm{OT}})$ *and*
- $\pi_{\mathrm{OT}}^{\mathsf{R}} \geq_{\text{Stat-UC}} \mathrm{IDEAL}(\mathcal{F}_{\mathrm{OT}})$ *if the receiver is honest and the sender is corrupted,*

where $\pi_{\mathrm{OT}}^{\mathsf{R}}$ is π_{OT} in the mode for statistical receiver security. Then, the protocol π_{nr2PC} in the $(\mathcal{F}_{\mathrm{CRS}}, \mathcal{F}_{\mathrm{CP}})$-hybrid model non-trivially long-term-realizes $\mathcal{F}_{\mathrm{2PC}}$ for non-reactive computations with long-term security for P_1.

Proof sketch. We begin the proof sketch with constructions of simulators for the dummy adversary.

[26]Unfortunately, it is not possible to reuse Construction 6 directly. This is due to the fact that we need to use the *same* instances of $\mathcal{F}_{\mathrm{CP}}$ in π_{nr2PC} and the OT protocol.

P_1 **Corrupted and** P_2 **Honest.** If P_1 is corrupted and P_2 is honest, the simulator extracts the correct input of P_1 and, in the long-term simulation, receives the correct input of P_2. Thus, real and ideal execution are identically distributed.

With respect to the non-long-term emulation, the circuit is garbled (and evaluated) with a possibly incorrect input for P_2. However, by sending an appropriate mask s_1, P_1 reconstructs the correct result. Due to the privacy property of the garbling scheme, the simulation is computationally indistinguishable from the real execution.

Definition 4.30 (Simulator for the Dummy Adversary)**.**

1. Extract P_1's input x_1 via P_1's input to $\mathcal{F}_{\mathrm{CP}}$ and send $(\texttt{input}, \mathit{sid}, x_1)$ to $\mathcal{F}_{\mathrm{lt2PC}}$ on behalf of P_1.
2. When receiving $(\texttt{input-provided}, \mathit{sid}, P_2)$ from $\mathcal{F}_{\mathrm{lt2PC}}$, send $(\texttt{lt-query}, \mathit{sid})$ to $\mathcal{F}_{\mathrm{lt2PC}}$ and receive answer $(\texttt{lt-query}, \mathit{sid}, (x_1'^1, x_2'^1), (y_1'^1, y_2'^1))$.
3. Continue the protocol execution like an honest P_2 would, using $x_2'^1$ as its input. Appropriately continue to simulate instances of $\mathcal{F}_{\mathrm{CP}}$.
4. After receiving y_2' from P_1, send $s_1 \oplus f(x_1'^1, x_2'^1)_1 \oplus y_1'^1$ to P_1 and simulate the corresponding message from $\mathcal{F}_{\mathrm{CP}}$.
5. On subsequent activations, allow a possible output from $\mathcal{F}_{\mathrm{lt2PC}}$ for P_2.

P_1 **Honest and** P_2 **Corrupted.** If P_1 is honest and P_2 is corrupted, the simulator extracts the correct input of P_2, but does not know the correct input of P_1. However, if the OT protocol has appropriate properties, namely long-term security for P_1, the simulated view is statistically indistinguishable. In particular, the view is independent of P_1's input until P_2 obtains the output of the garbled circuit's evaluation. However, this output is simulated appropriately and does not affect the indistinguishability.

Definition 4.31 (Simulator for the Dummy Adversary)**.**

1. Extract P_2's input x_2 as well as s_1, s_2 via P_2's input to $\mathcal{F}_{\mathrm{CP}}$ and send $(\texttt{input}, \mathit{sid}, x_2)$ to $\mathcal{F}_{\mathrm{lt2PC}}$ on behalf of P_2.
2. When receiving $(\texttt{input-provided}, \mathit{sid}, P_2)$ from $\mathcal{F}_{\mathrm{lt2PC}}$, send $(\texttt{lt-query}, \mathit{sid})$ to $\mathcal{F}_{\mathrm{lt2PC}}$ and receive answer $(\texttt{lt-query}, \mathit{sid}, (x_1'^1, x_2'^1), (y_1'^1, y_2'^1))$.
3. When receiving $(\texttt{input-provided}, \mathit{sid}, P_1)$ from $\mathcal{F}_{\mathrm{lt2PC}}$, perform the protocol like an honest P_1 would, using random choice bits for the OT protocol.
4. Send $y_2' = y_2'^1 \oplus s_2$ to P_2 and simulate the corresponding message from $\mathcal{F}_{\mathrm{CP}}$.
5. When receiving s_1 from P_2 and the corresponding message from $\mathcal{F}_{\mathrm{CP}}$, allow a possible output from $\mathcal{F}_{\mathrm{lt2PC}}$ for P_1.

Both Parties Honest. This case is very similar to the first case (P_2 honest and P_1 corrupted), except that the simulator also plays the honest party P_1, using random choice bits in the sessions of π_{OT}. In order to simulate y_2', the simulator sends $y_2' = y_2'^1 \oplus s_2$. For s_1, the value chosen by the internally emulated honest party P_1 is used. □

Again, we obtain a protocol with non-trivial long-term Rewinding UC security for P_1 in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model by replacing $\mathcal{F}_{\mathrm{CP}}$ with an appropriate protocol.

Reactive Two-Party Computation. The above strategy can be extended to *reactive* secure function evaluation using a folklore technique (used *e.g.* in [BKM+21]). To this end, let $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be an EUF-CMA-secure signature scheme with a deterministic[27] and length-regular[28] signing algorithm.

First, P_1 and P_2 use π_{nr2PC} to obtain a public key pk of Σ, along with shares sk_i of the secret key sk and a dummy state state_i for P_i. In subsequent rounds, P_1 and P_2 use pk, their share sk_i of the secret key, their share of the previous round's state as well as a signature of the previous state's share under sk as input, together with their actual input x_i^j. As output, they receive y_i^j, as well as a share of the state state_j and a signature on the state. By signing the state's share together with the current round number as well as the number i of the party, share re-use of a corrupted party is prevented.

In more detail, the evaluation is performed as follows.

1. Upon their first activation, P_1 and P_2 use π_{nr2PC} to evaluate the function f that
   - takes as input a tuple $((r_1^0, r_1^1), (r_2^0, r_2^1))$,
   - computes $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa; r_1^0 \oplus r_2^0)$,
   - $sk_1 = r_1^1 \oplus r_2^1$, $sk_2 = sk_1 \oplus sk$,
   - $\mathrm{state}_1 = \mathrm{state}_2 = 0^\kappa$ and
   - returns $(pk, sk_i, \sigma_i = \mathsf{Sign}(sk, (i||0||\mathrm{state}_i)), \mathrm{state}_i)$ for $i = 1, 2$.

   where r_i^0, r_i^1 are sampled uniformly at random. After evaluating f, the parties recover $pk, sk_i^0 = sk_i, \sigma_i^0 = \sigma_i, \mathrm{state}_i^0 = \mathrm{state}_i$.
2. To perform the j-th input evaluation round with $j \geq 1$, *i.e.* to evaluate f_j on input x_i^j, P_1 and P_2 use π_{nr2PC} to evaluate the function g_j that
   - takes as input a tuple $(pk_i', x_i', \mathrm{state}_i', \sigma_i', sk_i', r_i')$ for $i = 1, 2$,
   - returns a special error symbol $\perp$ if $pk_1' \neq pk_2'$,
   - returns a special error symbol $\perp$ if $\mathsf{Verify}(pk_1', (i||j-1||\mathrm{state}_i'), \sigma_i') \neq 1$ for $i = 1$ or $i = 2$,
   - otherwise, evaluates f_j on input $(x_1', x_2', \mathrm{state}_1' \oplus \mathrm{state}_2')$, resulting in $(y_1, y_2, \mathrm{state}'')$,

[27]Assuming a deterministic signing algorithm simplifies the presentation, but is not required.

[28]A deterministic signing algorithm is *length-regular* if for every $\kappa \in \mathbb{N}$, every $m_0, m_1 \in M$ and every $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$, it holds that if $|m_0| = |m_1|$, then $|\sigma_0 = \mathsf{Sign}(sk, m_0)| = |\sigma_1 = \mathsf{Sign}(sk, m_1)|$.

- sets $\text{state}''_1 = r'_1 \oplus r'_2$, $\text{state}''_2 = \text{state}''_1 \oplus \text{state}''$,
- computes signatures $\sigma''_i = \mathsf{Sign}(sk'_1 \oplus sk'_2, (i||j||\text{state}''_i))$ for $i = 1, 2$ and
- returns $(y_i, \text{state}''_i, \sigma''_i)$ as result for P_i.

To evaluate g_j, P_i uses $pk' = pk, sk'_i = sk_i, \sigma'_i = \sigma_i^{j-1}$ and $\text{state}'_i = \text{state}_i^{j-1}$ as well as uniformly random r_i of appropriate length for r'_i. After evaluating g_j, the parties recover y_i^j, state_i^j and σ_i^j. As result of the j-th round, P_i returns y_i^j.

Let π_{rSFE} denote the above protocol. We can then state the following proposition:

Proposition 4.11. *Let π_{OT} be a protocol in the $\mathcal{F}_{\text{CRS}}$-hybrid model such that*

- $\pi_{\text{OT}} \geq_{\text{UC}} \text{IDEAL}(\mathcal{F}_{\text{OT}})$ *and*
- $\pi_{\text{OT}}^{\mathsf{R}} \geq_{\text{Stat-UC}} \text{IDEAL}(\mathcal{F}_{\text{OT}})$ *if the receiver is honest and the sender is corrupted,*

where $\pi_{\text{OT}}^{\mathsf{R}}$ is π_{OT} in the mode for statistical receiver security. Then, the protocol π_{rSFE} in the $(\mathcal{F}_{\text{CRS}}, \mathcal{F}_{\text{CP}})$-hybrid model non-trivially long-term-realizes $\mathcal{F}_{\text{2PC}}$ for reactive computations with long-term security for P_1.

By replacing $\mathcal{F}_{\text{CP}}$ with an appropriate protocol, we obtain a protocol that non-trivially long-term-Rewinding-UC-realizes $\mathcal{F}_{\text{2PC}}$ for reactive computations with long-term security for P_1 in the $\mathcal{F}_{\text{CRS}}$-hybrid model.

We leave a detailed protocol description as well as a security proof for future work.

5. Conclusion and Outlook

In today's highly connected world, it is very important that cryptographic protocols remain secure even when they are executed alongside other protocols in a possibly adversarial scheduling. While the notion of Universal Composability (UC) provides very strong composable security guarantees, it suffers from several drawbacks. In this thesis, we presented two new frameworks and notions for composable secure multi-party computation (MPC) along with protocols for important tasks satisfying these notions.

Environmentally Friendly Composable Multi-Party Computation in the Plain Model from Standard (Timed) Assumptions

As our first contribution, we provided a novel solution for the important problem of attaining composable security without the use of setups.

By combining standard (non-timed) hardness assumptions with standard timed hardness assumptions, we were able to achieve composable MPC in the plain model with important properties not provided by any previous notion. In particular, our notion *Time-Lock UC (TLUC) security* is fully environmentally friendly and thus does not negatively affect polynomial-time game-based properties of other protocols. Featuring full UC reusability, we can take the best UC-secure protocol for a given task and realize its setup within TLUC, leading to a protocol in the plain model that is secure under composition. This property is only possible due to our novel simulation technique and not generally possible with previous approaches.

We constructed a TLUC-secure composable commitment scheme from standard and standard timed assumptions that is constant-round and uses its building blocks in a black-box way. Using this commitment scheme, we presented a protocol for constant-round composable general MPC in the plain model.

Outlook

The first result of this thesis raises a number of possible research questions.

Environmental Friendliness for Timed Properties. The established notion of environmental friendliness is only applicable to *non-timed* game-based properties. An open question is how environmental friendliness can be generalized to also capture *timed* game-based properties, *e.g.* the timed hiding property offered by timed commitment schemes. As a second step, it remains to investigate to what extent such a new notion is satisfied by TLUC security.

Weaker Hardness Assumptions. Apart from timed commitment schemes, our constructions rely on the specific assumption of trapdoor permutations with dense public description. We currently investigate if similar results can be achieved *e.g.* with (stand-alone-secure) oblivious transfer instead.

Complexity Leveraging. Finally, we currently work on adapting our protocol to the *shielded oracles* security notion combined with *complexity leveraging*, *i.e.* a setting where the simulation is performed with a (large) complexity advantage. In such a setting, we can replace timed assumptions with better-studied non-timed ones.

Historically, complexity leveraging suffers from the problem of correctly choosing the involved security levels. If the "low" security level (that can be broken by the simulator) turns out to be *too* low, it leads to protocols becoming completely insecure. Conversely, if the "low" security level is chosen too closely to the "high" one (that should not be broken by the simulator), it is conceivable that there is no gap whatsoever and the simulator is able to break both levels, negatively affecting security.

We intend to replace the timed commitment scheme in our constructions with a (non-timed) commitment scheme that is extractable with appropriate resources. Given that our proposed composable commitment scheme retains its security even when the timed commitment eventually loses its hiding property (as long as the coin-toss has been completed before), there is, intuitively, little risk involved when using a very low security level for the extractable commitment scheme. Conversely, due to the use of shielded oracles (that "shield" the extraction power), possible negative consequences of the levels being too close are also reduced in our proposed approach, as the values extracted by the shielded oracles are never directly observed, unless the commitment in question is opened.

Updatable Composable Security

With the availability of universal quantum computers, many of today's widely used cryptographic hardness assumptions such as RSA or the discrete logarithm problem will become invalid. Thus, it is important to start considering this problem *today*, by designing protocols with appropriate security guarantees, possibly including the ability to later update to a new cryptographic hardness assumption. Until now, no general security notion for this setting existed, neither with nor without composability.

By carefully incorporating rewinding into the UC execution, we obtained the first commitment scheme in the CRS-hybrid model that is both composable and statistically hiding at the same time, circumventing the long-standing impossibility results of Müller-Quade and Unruh [MU10]. The resulting commitment scheme not only has optimal asymptotic round complexity, but also requires standard assumptions only.

By extending the model to provide the environment with *complexity oracles*, we obtained a security notion that captures a setting where cryptographic hardness assumptions may become invalid at any time, *i.e.* before, during and after protocol execution. We showed that in even such a challenging setting, meaningful composable security guarantees can be obtained. In particular, we constructed a composable commitment scheme with a statistical hiding property whose binding property can be *updated* and proved that it realizes our newly introduced ideal functionality for updatable commitments.

We also gave an impossibility result for composable two-party computation with long-term security for both parties, namely that composable oblivious transfer with long-term security for both parties is already impossible in the considered setting. However, we constructed composable oblivious transfer with long-term security for *one* party, which is the best guarantee in our setting. Using this protocol for oblivious transfer, we sketched how composable (reactive) general two-party computation with long-term security for one party can be achieved. For the special cases of composable zero-knowledge and composable commit-and-proof, we provided constructions with long-term security for both parties.

Outlook

We intend to investigate the following open problems.

Constructions in the Plain Model. Given the drawbacks of trusted setups and the fact that the extraction technique we built upon allows composable MPC in the plain model (albeit known results do not achieve long-term security), a natural question to ask is whether our constructions can be adapted accordingly, yielding a notion for composable long-term security in the plain model. In such a setting, equivocation via the CRS is not possible anymore, requiring a more complex construction as well as an even more involved security proof.

Updatable Security for Other Tasks. Apart from commitment schemes, it remains to investigate which other tasks can be achieved with meaningful (composable) updatable security. Natural candidates are "long-living" tasks, *e.g.* commit-and-prove as well as general two-party computations.

Bibliography

[ABBC10] T. Acar, M. Belenkiy, M. Bellare, and D. Cash. "Cryptographic Agility and Its Relation to Circular Encryption". In: *Advances in Cryptology – EUROCRYPT 2010*. Ed. by H. Gilbert. Vol. 6110. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2010, pp. 403–422. DOI: 10.1007/978-3-642-13190-5_21.

[AGH+19] D. Achenbach, R. Gröll, T. Hackenjos, A. Koch, B. Löwe, J. Mechler, J. Müller-Quade, and J. Rill. "Your Money or Your Life - Modeling and Analyzing the Security of Electronic Payment in the UC Framework". In: *FC 2019: 23rd International Conference on Financial Cryptography and Data Security*. Ed. by I. Goldberg and T. Moore. Vol. 11598. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2019, pp. 243–261. DOI: 10.1007/978-3-030-32101-7_16.

[AHLR18] G. Asharov, S. Halevi, Y. Lindell, and T. Rabin. "Privacy-Preserving Search of Similar Patients in Genomic Data". In: *Proc. Priv. Enhancing Technol.* 2018.4 (2018), pp. 104–124. DOI: 10.1515/popets-2018-0034. URL: https://doi.org/10.1515/popets-2018-0034.

[AL17] G. Asharov and Y. Lindell. "A Full Proof of the BGW Protocol for Perfectly Secure Multiparty Computation". In: *Journal of Cryptology* 30.1 (2017), pp. 58–151. DOI: 10.1007/s00145-015-9214-4.

[B81] M. Blum. "Coin Flipping by Telephone". In: *Advances in Cryptology – CRYPTO'81*. Ed. by A. Gersho. Vol. ECE Report 82-04. U.C. Santa Barbara, Dept. of Elec. and Computer Eng., 1981, pp. 11–15.

[B97] D. Beaver. "Commodity-Based Cryptography (Extended Abstract)". In: *29th Annual ACM Symposium on Theory of Computing*. ACM Press, 1997, pp. 446–455. DOI: 10.1145/258533.258637.

[BBK+23] R. Berger, B. Broadnax, M. Klooß, J. Mechler, J. Müller-Quade, A. Ottenhues, and M. Raiber. "Composable Long-Term Security with Rewinding". In: *Theory of Cryptography - 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 - December 2, 2023, Proceedings, Part IV*. Ed. by G. N. Rothblum and H. Wee. Vol. 14372. Lecture Notes in Computer Science. Springer, 2023, pp. 510–541. DOI: 10.1007/978-3-031-48624-1_19. URL: https://doi.org/10.1007/978-3-031-48624-1%5C_19.

[BCD$^+$09] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. I. Schwartzbach, and T. Toft. "Secure Multiparty Computation Goes Live". In: *FC 2009: 13th International Conference on Financial Cryptography and Data Security.* Ed. by R. Dingledine and P. Golle. Vol. 5628. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2009, pp. 325–343.

[BCH$^+$20] C. Badertscher, R. Canetti, J. Hesse, B. Tackmann, and V. Zikas. "Universal Composition with Global Subroutines: Capturing Global Setup Within Plain UC". In: *TCC 2020: 18th Theory of Cryptography Conference, Part III.* Ed. by R. Pass and K. Pietrzak. Vol. 12552. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2020, pp. 1–30. DOI: 10.1007/978-3-030-64381-2_1.

[BDD$^+$20] C. Baum, B. David, R. Dowsley, J. B. Nielsen, and S. Oechsner. "CRAFT: Composable Randomness and Almost Fairness from Time". Cryptology ePrint Archive, Report 2020/784. https://eprint.iacr.org/2020/784. 2020.

[BDD$^+$21] C. Baum, B. David, R. Dowsley, J. B. Nielsen, and S. Oechsner. "TARDIS: A Foundation of Time-Lock Puzzles in UC". In: *Advances in Cryptology – EUROCRYPT 2021, Part III.* Ed. by A. Canteaut and F.-X. Standaert. Vol. 12698. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2021, pp. 429–459. DOI: 10.1007/978-3-030-77883-5_15.

[BDH$^+$17] B. Broadnax, N. Döttling, G. Hartung, J. Müller-Quade, and M. Nagel. "Concurrently Composable Security with Shielded Super-Polynomial Simulators". In: *Advances in Cryptology – EUROCRYPT 2017, Part I.* Ed. by J.-S. Coron and J. B. Nielsen. Vol. 10210. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2017, pp. 351–381. DOI: 10.1007/978-3-319-56620-7_13.

[BDH$^+$21] W. Beskorovajnov, F. Dörre, G. Hartung, A. Koch, J. Müller-Quade, and T. Strufe. "ConTra Corona: Contact Tracing against the Coronavirus by Bridging the Centralized-Decentralized Divide for Stronger Privacy". In: *Advances in Cryptology – ASIACRYPT 2021, Part II.* Ed. by M. Tibouchi and H. Wang. Vol. 13091. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2021, pp. 665–695. DOI: 10.1007/978-3-030-92075-3_23.

[BFG19] J. Brendel, M. Fischlin, and F. Günther. "Breakdown Resilience of Key Exchange Protocols: NewHope, TLS 1.3, and Hybrids". In: *ESORICS 2019: 24th European Symposium on Research in Computer Security, Part II.* Ed. by K. Sako, S. Schneider, and P. Y. A. Ryan. Vol. 11736. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2019, pp. 521–541. DOI: 10.1007/978-3-030-29962-0_25.

[BFMR18] B. Broadnax, V. Fetzer, J. Müller-Quade, and A. Rupp. “Non-malleability vs. CCA-Security: The Case of Commitments”. In: *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II.* Ed. by M. Abdalla and R. Dahab. Vol. 10770. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2018, pp. 312–337. DOI: 10.1007/978-3-319-76581-5_11.

[BGB17] A. Buldas, M. Geihs, and J. A. Buchmann. “Long-Term Secure Commitments via Extractable-Binding Commitments”. In: *ACISP 17: 22nd Australasian Conference on Information Security and Privacy, Part I.* Ed. by J. Pieprzyk and S. Suriadi. Vol. 10342. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2017, pp. 65–81.

[BGR$^+$15] H. Brenner, V. Goyal, S. Richelson, A. Rosen, and M. Vald. “Fast Non-Malleable Commitments”. In: *ACM CCS 2015: 22nd Conference on Computer and Communications Security.* Ed. by I. Ray, N. Li, and C. Kruegel. ACM Press, 2015, pp. 1048–1057. DOI: 10.1145/2810103.2813721.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. “Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)”. In: *20th Annual ACM Symposium on Theory of Computing.* ACM Press, 1988, pp. 1–10. DOI: 10.1145/62212.62213.

[BHR12] M. Bellare, V. T. Hoang, and P. Rogaway. “Foundations of garbled circuits”. In: *ACM CCS 2012: 19th Conference on Computer and Communications Security.* Ed. by T. Yu, G. Danezis, and V. D. Gligor. ACM Press, 2012, pp. 784–796. DOI: 10.1145/2382196.2382279.

[BKM$^+$21] B. Broadnax, A. Koch, J. Mechler, T. Müller, J. Müller-Quade, and M. Nagel. “Fortified Multi-Party Computation: Taking Advantage of Simple Secure Hardware Modules”. In: *Proceedings on Privacy Enhancing Technologies* 2021.4 (2021), pp. 312–338. DOI: 10.2478/popets-2021-0072.

[BM82] M. Blum and S. Micali. “How to Generate Cryptographically Strong Sequences of Pseudo Random Bits”. In: *23rd Annual Symposium on Foundations of Computer Science.* IEEE Computer Society Press, 1982, pp. 112–117. DOI: 10.1109/SFCS.1982.72.

[BMM21] B. Broadnax, J. Mechler, and J. Müller-Quade. “Environmentally Friendly Composable Multi-party Computation in the Plain Model from Standard (Timed) Assumptions”. In: *TCC 2021: 19th Theory of Cryptography Conference, Part I.* Ed. by K. Nissim and B. Waters. Vol. 13042. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2021, pp. 750–781. DOI: 10.1007/978-3-030-90459-3_25.

[BN00] D. Boneh and M. Naor. “Timed Commitments”. In: *Advances in Cryptology – CRYPTO 2000.* Ed. by M. Bellare. Vol. 1880. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2000, pp. 236–254. DOI: 10.1007/3-540-44598-6_15.

[BR94] M. Bellare and P. Rogaway. “Entity Authentication and Key Distribution”. In: *Advances in Cryptology – CRYPTO’93*. Ed. by D. R. Stinson. Vol. 773. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 1994, pp. 232–249. DOI: 10.1007/3-540-48329-2_21.

[BS05] B. Barak and A. Sahai. “How To Play Almost Any Mental Game Over The Net - Concurrent Composition via Super-Polynomial Simulation”. In: *46th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 2005, pp. 543–552. DOI: 10.1109/SFCS.2005.43.

[C01] R. Canetti. “Universally Composable Security: A New Paradigm for Cryptographic Protocols”. In: *42nd Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 2001, pp. 136–145. DOI: 10.1109/SFCS.2001.959888.

[C20] R. Canetti. “Universally Composable Security”. In: *J. ACM* 67.5 (2020), 28:1–28:94. DOI: 10.1145/3402457. URL: https://doi.org/10.1145/3402457.

[CDG$^+$15] D. Cabarcas, D. Demirel, F. Göpfert, J. Lancrenon, and T. Wunderer. “An Unconditionally Hiding and Long-Term Binding Post-Quantum Commitment Scheme”. Cryptology ePrint Archive, Report 2015/628. https://eprint.iacr.org/2015/628. 2015.

[CDPW06] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. “Universally Composable Security with Global Setup”. Cryptology ePrint Archive, Report 2006/432. https://eprint.iacr.org/2006/432. 2006.

[CDPW07] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. “Universally Composable Security with Global Setup”. In: *TCC 2007: 4th Theory of Cryptography Conference*. Ed. by S. P. Vadhan. Vol. 4392. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2007, pp. 61–85. DOI: 10.1007/978-3-540-70936-7_4.

[CF01] R. Canetti and M. Fischlin. “Universally Composable Commitments”. In: *Advances in Cryptology – CRYPTO 2001*. Ed. by J. Kilian. Vol. 2139. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2001, pp. 19–40. DOI: 10.1007/3-540-44647-8_2.

[CKL03] R. Canetti, E. Kushilevitz, and Y. Lindell. “On the Limitations of Universally Composable Two-Party Computation without Set-up Assumptions”. In: *Advances in Cryptology – EUROCRYPT 2003*. Ed. by E. Biham. Vol. 2656. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2003, pp. 68–86. DOI: 10.1007/3-540-39200-9_5.

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. “Universally composable two-party and multi-party secure computation”. In: *34th Annual ACM Symposium on Theory of Computing*. ACM Press, 2002, pp. 494–503. DOI: 10.1145/509907.509980.

[CLP10] R. Canetti, H. Lin, and R. Pass. “Adaptive Hardness and Composable Security in the Plain Model from Standard Assumptions”. In: *51st Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 2010, pp. 541–550. DOI: 10.1109/FOCS.2010.86.

[CLP13a] R. Canetti, H. Lin, and R. Pass. “From Unprovability to Environmentally Friendly Protocols”. In: *54th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 2013, pp. 70–79. DOI: 10.1109/FOCS.2013.16.

[CLP13b] R. Canetti, H. Lin, and R. Pass. “From Unprovability to Environmentally Friendly Protocols, full version”. 2013. URL: https://www.cs.cornell.edu/~rafael/papers/EnvFriendly-proc.pdf.

[CLP16] R. Canetti, H. Lin, and R. 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.

[CR03] R. Canetti and T. Rabin. “Universal Composition with Joint State”. In: *Advances in Cryptology – CRYPTO 2003*. Ed. by D. Boneh. Vol. 2729. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2003, pp. 265–281. DOI: 10.1007/978-3-540-45146-4_16.

[DDN00] D. Dolev, C. Dwork, and M. Naor. “Nonmalleable Cryptography”. In: *SIAM Journal on Computing* 30.2 (2000), pp. 391–437.

[DIO98] G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. “Non-Interactive and Non-Malleable Commitment”. In: *30th Annual ACM Symposium on Theory of Computing*. ACM Press, 1998, pp. 141–150. DOI: 10.1145/276698.276722.

[DKM$^+$22] N. Döttling, A. Koch, S. Maier, J. Mechler, J. Müller-Quade, and M. Tiepelt. “Towards Everlasting Bit Commitment from Quantum Decoherence”. unpublished. 2022.

[DKM11] N. Döttling, D. Kraschewski, and J. Müller-Quade. “Unconditional and Composable Security Using a Single Stateful Tamper-Proof Hardware Token”. In: *TCC 2011: 8th Theory of Cryptography Conference*. Ed. by Y. Ishai. Vol. 6597. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2011, pp. 164–181. DOI: 10.1007/978-3-642-19571-6_11.

[DL15] D. Demirel and J. Lancrenon. “How to Securely Prolong the Computational Bindingness of Pedersen Commitments”. Cryptology ePrint Archive, Report 2015/584. https://eprint.iacr.org/2015/584. 2015.

[DMM22] D. Doerner, J. Mechler, and J. Müller-Quade. “Hardening the Security of Server-Aided MPC Using Remotely Unhackable Hardware Modules”. In: *GI Sicherheit*. 2022.

[DMM23] F. Dörre, J. Mechler, and J. Müller-Quade. “Practically Efficient Private Set Intersection from Trusted Hardware with Side-Channels”. In: *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part IV*. Ed. by J. Guo and R. Steinfeld. Vol. 14441. Lecture Notes in Computer Science. Springer, 2023, pp. 268–301. DOI: 10.1007/978-981-99-8730-6_9. URL: https://doi.org/10.1007/978-981-99-8730-6%5C_9.

[DMRV13] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Venkitasubramaniam. “Adaptive and Concurrent Secure Computation from New Adaptive, Non-malleable Commitments”. In: *Advances in Cryptology – ASIACRYPT 2013, Part I*. Ed. by K. Sako and P. Sarkar. Vol. 8269. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2013, pp. 316–336. DOI: 10.1007/978-3-642-42033-7_17.

[DN02] I. Damgård and J. B. Nielsen. “Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor”. In: *Advances in Cryptology – CRYPTO 2002*. Ed. by M. Yung. Vol. 2442. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2002, pp. 581–596. DOI: 10.1007/3-540-45708-9_37.

[DP92] A. De Santis and G. Persiano. “Zero-Knowledge Proofs of Knowledge Without Interaction (Extended Abstract)”. In: *33rd Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 1992, pp. 427–436. DOI: 10.1109/SFCS.1992.267809.

[DS13] I. Damgård and A. Scafuro. “Unconditionally Secure and Universally Composable Commitments from Physical Assumptions”. In: *Advances in Cryptology – ASIACRYPT 2013, Part II*. Ed. by K. Sako and P. Sarkar. Vol. 8270. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2013, pp. 100–119. DOI: 10.1007/978-3-642-42045-0_6.

[E84] T. ElGamal. “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms”. In: *Advances in Cryptology – CRYPTO’84*. Ed. by G. R. Blakley and D. Chaum. Vol. 196. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 1984, pp. 10–18.

[EFKP20] N. Ephraim, C. Freitag, I. Komargodski, and R. Pass. “Non-Malleable Time-Lock Puzzles and Applications”. Tech. rep. 2020.

[FF11] M. Fischlin and R. Fischlin. “Efficient Non-Malleable Commitment Schemes”. In: *Journal of Cryptology* 24.1 (2011), pp. 203–244. DOI: 10.1007/s00145-009-9043-4.

[FKPS21] C. Freitag, I. Komargodski, R. Pass, and N. Sirkin. “Non-malleable Time-Lock Puzzles and Applications”. In: *TCC 2021: 19th Theory of Cryptography Conference, Part III*. Ed. by K. Nissim and B. Waters. Vol. 13044. Lecture

Notes in Computer Science. Springer, Heidelberg, Germany, 2021, pp. 447–479. DOI: 10.1007/978-3-030-90456-2_15.

[G01] O. Goldreich. "Foundations of Cryptography: Basic Tools". Vol. 1. Cambridge, UK: Cambridge University Press, 2001, pp. xix + 372.

[G04] O. Goldreich. "Foundations of Cryptography: Basic Applications". Vol. 2. Cambridge, UK: Cambridge University Press, 2004.

[G08] O. Goldreich. "Computational complexity - a conceptual perspective". Cambridge University Press, 2008. DOI: 10.1017/CBO9780511804106. URL: https://doi.org/10.1017/CBO9780511804106.

[G96] L. K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *28th Annual ACM Symposium on Theory of Computing*. ACM Press, 1996, pp. 212–219. DOI: 10.1145/237814.237866.

[GGJS12] S. Garg, V. Goyal, A. Jain, and A. Sahai. "Concurrently Secure Computation in Constant Rounds". In: *Advances in Cryptology – EUROCRYPT 2012*. Ed. by D. Pointcheval and T. Johansson. Vol. 7237. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2012, pp. 99–116. DOI: 10.1007/978-3-642-29011-4_8.

[GK16] S. Goldwasser and Y. T. Kalai. "Cryptographic Assumptions: A Position Paper". In: *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*. Ed. by E. Kushilevitz and T. Malkin. Vol. 9562. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2016, pp. 505–522. DOI: 10.1007/978-3-662-49096-9_21.

[GK90] O. Goldreich and H. Krawczyk. "On the Composition of Zero-Knowledge Proof Systems". In: *Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, UK, July 16-20, 1990, Proceedings*. Ed. by M. Paterson. Vol. 443. Lecture Notes in Computer Science. Springer, 1990, pp. 268–282. DOI: 10.1007/BFb0032038. URL: https://doi.org/10.1007/BFb0032038.

[GKP18] S. Garg, S. Kiyoshima, and O. Pandey. "A New Approach to Black-Box Concurrent Secure Computation". In: *Advances in Cryptology – EUROCRYPT 2018, Part II*. Ed. by J. B. Nielsen and V. Rijmen. Vol. 10821. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2018, pp. 566–599. DOI: 10.1007/978-3-319-78375-8_19.

[GLP$^+$12] V. Goyal, H. Lin, O. Pandey, R. Pass, and A. Sahai. "Round-Efficient Concurrently Composable Secure Computation via a Robust Extraction Lemma". Cryptology ePrint Archive, Report 2012/652. https://eprint.iacr.org/2012/652. 2012.

[GLP$^+$15] V. Goyal, H. Lin, O. Pandey, R. Pass, and A. Sahai. “Round-Efficient Concurrently Composable Secure Computation via a Robust Extraction Lemma”. In: *TCC 2015: 12th Theory of Cryptography Conference, Part I.* Ed. by Y. Dodis and J. B. Nielsen. Vol. 9014. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2015, pp. 260–289. DOI: 10.1007/978-3-662-46494-6_12.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. “How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority”. In: *19th Annual ACM Symposium on Theory of Computing.* Ed. by A. Aho. ACM Press, 1987, pp. 218–229. DOI: 10.1145/28395.28420.

[GMY03] J. A. Garay, P. D. MacKenzie, and K. Yang. “Strengthening Zero-Knowledge Protocols Using Signatures”. In: *Advances in Cryptology – EUROCRYPT 2003.* Ed. by E. Biham. Vol. 2656. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2003, pp. 177–194. DOI: 10.1007/3-540-39200-9_11.

[GRRV14] V. Goyal, S. Richelson, A. Rosen, and M. Vald. “An Algebraic Approach to Non-malleability”. In: *55th Annual Symposium on Foundations of Computer Science.* IEEE Computer Society Press, 2014, pp. 41–50. DOI: 10.1109/FOCS.2014.13.

[GVW15] S. Gorbunov, V. Vaikuntanathan, and D. Wichs. “Leveled Fully Homomorphic Signatures from Standard Lattices”. In: *47th Annual ACM Symposium on Theory of Computing.* Ed. by R. A. Servedio and R. Rubinfeld. ACM Press, 2015, pp. 469–477. DOI: 10.1145/2746539.2746576.

[HHR16] J. Hesse, D. Hofheinz, and A. Rupp. “Reconfigurable Cryptography: A Flexible Approach to Long-Term Security”. In: *TCC 2016-A: 13th Theory of Cryptography Conference, Part I.* Ed. by E. Kushilevitz and T. Malkin. Vol. 9562. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2016, pp. 416–445. DOI: 10.1007/978-3-662-49096-9_18.

[HV15] C. Hazay and M. Venkitasubramaniam. “On Black-Box Complexity of Universally Composable Security in the CRS Model”. In: *Advances in Cryptology – ASIACRYPT 2015, Part II.* Ed. by T. Iwata and J. H. Cheon. Vol. 9453. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2015, pp. 183–209. DOI: 10.1007/978-3-662-48800-3_8.

[HW09] S. Hohenberger and B. Waters. “Short and Stateless Signatures from the RSA Assumption”. In: *Advances in Cryptology – CRYPTO 2009.* Ed. by S. Halevi. Vol. 5677. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2009, pp. 654–670. DOI: 10.1007/978-3-642-03356-8_38.

[IPS08] Y. Ishai, M. Prabhakaran, and A. Sahai. “Founding Cryptography on Oblivious Transfer - Efficiently”. In: *Advances in Cryptology – CRYPTO 2008.* Ed. by D. Wagner. Vol. 5157. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2008, pp. 572–591. DOI: 10.1007/978-3-540-85174-5_32.

[K05] V. Kolesnikov. "Gate Evaluation Secret Sharing and Secure One-Round Two-Party Computation". In: *Advances in Cryptology – ASIACRYPT 2005*. Ed. by B. K. Roy. Vol. 3788. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2005, pp. 136–155. DOI: 10.1007/11593447_8.

[K06] V. Klima. "Tunnels in Hash Functions: MD5 Collisions Within a Minute". Cryptology ePrint Archive, Report 2006/105. https://eprint.iacr.org/2006/105. 2006.

[K14] S. Kiyoshima. "Round-Efficient Black-Box Construction of Composable Multi-Party Computation". In: *Advances in Cryptology – CRYPTO 2014, Part II*. Ed. by J. A. Garay and R. Gennaro. Vol. 8617. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2014, pp. 351–368. DOI: 10.1007/978-3-662-44381-1_20.

[K20] S. Kiyoshima. "Statistical Concurrent Non-Malleable Zero-Knowledge from One-Way Functions". In: *Journal of Cryptology* 33.3 (2020), pp. 1318–1361. DOI: 10.1007/s00145-020-09348-x.

[K88] J. Kilian. "Founding Cryptography on Oblivious Transfer". In: *20th Annual ACM Symposium on Theory of Computing*. ACM Press, 1988, pp. 20–31. DOI: 10.1145/62212.62215.

[KL11] D. Kidron and Y. Lindell. "Impossibility Results for Universal Composability in Public-Key Models and with Fixed Inputs". In: *Journal of Cryptology* 24.3 (2011), pp. 517–544. DOI: 10.1007/s00145-010-9069-7.

[KLP05] Y. T. Kalai, Y. Lindell, and M. Prabhakaran. "Concurrent general composition of secure protocols in the timing model". In: *37th Annual ACM Symposium on Theory of Computing*. Ed. by H. N. Gabow and R. Fagin. ACM Press, 2005, pp. 644–653. DOI: 10.1145/1060590.1060687.

[KLX20] J. Katz, J. Loss, and J. Xu. "On the Security of Time-Lock Puzzles and Timed Commitments". In: *TCC 2020: 18th Theory of Cryptography Conference, Part III*. Ed. by R. Pass and K. Pietrzak. Vol. 12552. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2020, pp. 390–413. DOI: 10.1007/978-3-030-64381-2_14.

[L03] Y. Lindell. "General Composition and Universal Composability in Secure Multi-Party Computation". In: *44th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 2003, pp. 394–403. DOI: 10.1109/SFCS.2003.1238213.

[L16] Y. Lindell. "How To Simulate It - A Tutorial on the Simulation Proof Technique". Cryptology ePrint Archive, Report 2016/046. https://eprint.iacr.org/2016/046. 2016.

[LPV09] H. Lin, R. Pass, and M. Venkitasubramaniam. “A unified framework for concurrent security: universal composability from stand-alone non-malleability”. In: *41st Annual ACM Symposium on Theory of Computing*. Ed. by M. Mitzenmacher. ACM Press, 2009, pp. 179–188. DOI: 10.1145/1536414.1536441.

[MMN18] J. Mechler, J. Müller-Quade, and T. Nilges. “Reusing Tamper-Proof Hardware in UC-Secure Protocols”. In: *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*. Ed. by M. Abdalla and R. Dahab. Vol. 10769. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2018, pp. 463–493. DOI: 10.1007/978-3-319-76578-5_16.

[MMSU22] B. Magri, G. Malavolta, D. Schröder, and D. Unruh. “Everlasting UC Commitments from Fully Malicious PUFs”. In: *J. Cryptol.* 35.3 (2022), p. 20.

[MMV13] M. Mahmoody, T. Moran, and S. P. Vadhan. “Publicly verifiable proofs of sequential work”. In: *ITCS 2013: 4th Innovations in Theoretical Computer Science*. Ed. by R. D. Kleinberg. Association for Computing Machinery, 2013, pp. 373–388. DOI: 10.1145/2422436.2422479.

[MPR06] S. Micali, R. Pass, and A. Rosen. “Input-Indistinguishable Computation”. In: *47th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 2006, pp. 367–378. DOI: 10.1109/FOCS.2006.43.

[MR11] U. Maurer and R. Renner. “Abstract Cryptography”. In: *ICS 2011: 2nd Innovations in Computer Science*. Ed. by B. Chazelle. Tsinghua University Press, 2011, pp. 1–21.

[MU10] J. Müller-Quade and D. Unruh. “Long-Term Security and Universal Composability”. In: *Journal of Cryptology* 23.4 (2010), pp. 594–671. DOI: 10.1007/s00145-010-9068-8.

[MY04] P. D. MacKenzie and K. Yang. “On Simulation-Sound Trapdoor Commitments”. In: *Advances in Cryptology – EUROCRYPT 2004*. Ed. by C. Cachin and J. Camenisch. Vol. 3027. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2004, pp. 382–400. DOI: 10.1007/978-3-540-24676-3_23.

[N03] J. Nielsen. “On Protocol Security in the Cryptographic Model”. English. PhD thesis. 2003.

[N90] M. Naor. “Bit Commitment Using Pseudo-Randomness”. In: *Advances in Cryptology – CRYPTO’89*. Ed. by G. Brassard. Vol. 435. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 1990, pp. 128–136. DOI: 10.1007/0-387-34805-0_13.

[N91] M. Naor. “Bit Commitment Using Pseudorandomness”. In: *Journal of Cryptology* 4.2 (1991), pp. 151–158. DOI: 10.1007/BF00196774.

[OOR+14] C. Orlandi, R. Ostrovsky, V. Rao, A. Sahai, and I. Visconti. “Statistical Concurrent Non-malleable Zero Knowledge”. In: *TCC 2014: 11th Theory of Cryptography Conference*. Ed. by Y. Lindell. Vol. 8349. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2014, pp. 167–191. DOI: 10.1007/978-3-642-54242-8_8.

[OP19] D. Ott and C. Peikert. “Identifying Research Challenges in Post Quantum Cryptography Migration and Cryptographic Agility”. In: *CoRR* abs/1909.07353 (2019). ARXIV-ID: 1909.07353. URL: http://arxiv.org/abs/1909.07353.

[OPV08] R. Ostrovsky, G. Persiano, and I. Visconti. “Constant-Round Concurrent Non-Malleable Commitments and Decommitments”. Cryptology ePrint Archive, Report 2008/235. https://eprint.iacr.org/2008/235. 2008.

[P03] R. Pass. “Simulation in Quasi-Polynomial Time, and Its Application to Protocol Composition”. In: *Advances in Cryptology – EUROCRYPT 2003*. Ed. by E. Biham. Vol. 2656. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2003, pp. 160–176. DOI: 10.1007/3-540-39200-9_10.

[P92] T. P. Pedersen. “Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing”. In: *Advances in Cryptology – CRYPTO’91*. Ed. by J. Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 1992, pp. 129–140. DOI: 10.1007/3-540-46766-1_9.

[PR05a] R. Pass and A. Rosen. “Concurrent Non-Malleable Commitments”. In: *46th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 2005, pp. 563–572. DOI: 10.1109/SFCS.2005.27.

[PR05b] R. Pass and A. Rosen. “New and improved constructions of non-malleable cryptographic protocols”. In: *37th Annual ACM Symposium on Theory of Computing*. Ed. by H. N. Gabow and R. Fagin. ACM Press, 2005, pp. 533–542. DOI: 10.1145/1060590.1060670.

[PR08] M. Prabhakaran and M. Rosulek. “Cryptographic Complexity of Multi-Party Computation Problems: Classifications and Separations”. In: *Advances in Cryptology – CRYPTO 2008*. Ed. by D. Wagner. Vol. 5157. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2008, pp. 262–279. DOI: 10.1007/978-3-540-85174-5_15.

[PRS02] M. Prabhakaran, A. Rosen, and A. Sahai. “Concurrent Zero Knowledge with Logarithmic Round-Complexity”. In: *43rd Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 2002, pp. 366–375. DOI: 10.1109/SFCS.2002.1181961.

[PS04] M. Prabhakaran and A. Sahai. “New notions of security: Achieving universal composability without trusted setup”. In: *36th Annual ACM Symposium on Theory of Computing*. Ed. by L. Babai. ACM Press, 2004, pp. 242–251. DOI: 10.1145/1007352.1007394.

[PTV14] R. Pass, W.-L. D. Tseng, and M. Venkitasubramaniam. “Concurrent Zero Knowledge, Revisited”. In: *Journal of Cryptology* 27.1 (2014), pp. 45–66. DOI: 10.1007/s00145-012-9137-2.

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

[PW09] R. Pass and H. Wee. “Black-Box Constructions of Two-Party Protocols from One-Way Functions”. In: *TCC 2009: 6th Theory of Cryptography Conference*. Ed. by O. Reingold. Vol. 5444. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2009, pp. 403–418. DOI: 10.1007/978-3-642-00457-5_24.

[R05] O. Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *37th Annual ACM Symposium on Theory of Computing*. Ed. by H. N. Gabow and R. Fagin. ACM Press, 2005, pp. 84–93. DOI: 10.1145/1060590.1060603.

[RSA78] R. L. Rivest, A. Shamir, and L. M. Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. In: *Communications of the Association for Computing Machinery* 21.2 (1978), pp. 120–126.

[RSW96] R. L. Rivest, A. Shamir, and D. A. Wagner. “Time-lock puzzles and timed-release crypto”. 1996.

[S94] P. W. Shor. “Algorithms for Quantum Computation: Discrete Logarithms and Factoring”. In: *35th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[SKP16] M. Stevens, P. Karpman, and T. Peyrin. “Freestart Collision for Full SHA-1”. In: *Advances in Cryptology – EUROCRYPT 2016, Part I*. Ed. by M. Fischlin and J.-S. Coron. Vol. 9665. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2016, pp. 459–483. DOI: 10.1007/978-3-662-49890-3_18.

[Y82] A. C.-C. Yao. “Theory and Applications of Trapdoor Functions (Extended Abstract)”. In: *23rd Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 1982, pp. 80–91. DOI: 10.1109/SFCS.1982.45.

[Y86] A. C.-C. Yao. “How to Generate and Exchange Secrets (Extended Abstract)”. In: *27th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 1986, pp. 162–167. DOI: 10.1109/SFCS.1986.25.

A. Appendix

The appendix contains technical elaborations not found in the main body of this thesis.

A.1. TLUC Security

In the following, we discuss aspects of the framework and notion that are not found in the main body. In particular, we provide formal definitions of legal adversaries and environments as well as how computation steps are counted.

Remark A.1. We have chosen to model TLUC security using the mechanisms available in the UC framework. However, it is conceivable to achieve a similar notion of security using different mechanisms. To this end, an anonymous reviewer has suggested the use of *shells*, which have seen heavy use in recent iterations of the UC paper to transparently enforce rules of the security notion.

However, there is no indication that resorting to shells would lead to easier definition or improve the notion's properties. Indeed, composition is limited because environments cannot internally execute simulators that break timed assumptions without affecting the number of counted steps. In any setting where the steps of the environment are counted correctly, this would affect timed assumptions of the challenge protocol, leading to limited composition.

A.1.1. Legal Adversaries

We first define *legal* adversaries, *i.e.* adversaries that correctly handle timer-related messages:

Definition A.1 (Legal Adversaries). An adversary $\mathcal{A}$ is called *legal* if

1. upon receiving $(\texttt{timer}, \mu, id, t)$ from some ITI with extended identity μ via the backdoor tape, $\mathcal{A}$ immediately sends $(\texttt{timeout}, \mu, id, t)$ to the environment once.

2. it does not send the message $(\texttt{timer}, \mu, id, t)$ to the environment if the party with extended ID μ is honest and has not sent $(\texttt{timer}, \mu, id, t)$ to the adversary.

3. upon receiving $(\texttt{notify}, \mu, id)$ from some ITI with extended identity μ via the backdoor tape, $\mathcal{A}$ immediately sends $(\texttt{notify}, \mu, id)$ to the environment once.

4. upon receiving $(\texttt{notify}, \mu, id, b)$ from the environment, $\mathcal{A}$ immediately delivers $(\texttt{notify}, \mu, id, b)$ to the backdoor tape of the ITI with extended identity μ once.

A.1.2. Legal Environments

The definition of legal environments is more involved and requires several auxiliary definitions and discussions, starting with how computation steps are counted.

Counting Computation Steps. When an honest party P sets up a timer with timeout ℓ, we want P to be able to learn when the timer has expired. To this end, it is first necessary to define how computation steps are counted against the timer by the environment, which is ultimately responsible to signal if a timeout has occurred.

Suppose that we want to prove that a protocol π using a timed assumption emulates a protocol ϕ. At some point in the reduction, we may have to construct a stand-alone adversary $\mathcal{A}'$ against the timed assumption that incorporates the whole TLUC execution and uses a distinguishing environment to contradict the security of the timed assumption. In this situation, the counting of steps by $\mathcal{Z}$ and by $\mathcal{A}'$ has to be compatible in the sense that if $\mathcal{Z}$ never triggers a timeout, neither does $\mathcal{A}'$ in its game.

While the intuition is clear, the definition must account for the fact that adversary and protocol under consideration might change, *e.g.* when considering protocol emulation. To this end, we explicitly parameterize environments with a protocol and adversary to make clear relative to which ones the number of performed steps is counted. Let $\mathcal{Z}[\pi, \mathcal{A}]$ denote the environment $\mathcal{Z}$ that expects to interact with protocol π and adversary $\mathcal{A}$ and counts its steps accordingly.

Note that an environment may not have enough information to precisely calculate the correct number of steps performed by other entities, *e.g.* if the protocol is probabilistic. Considering the security guarantee we want to capture, this is no problem as long as the environment does not *under-estimate* the number of steps performed. In particular, security is not affected if the number of steps performed is estimated too high, leading to a timeout triggered too early. This is accounted for in Definition A.5.

We also require environments to perform the calculation independent of the protocol parties' inputs and outputs. (For a discussion, see Remark A.3.) Otherwise, we would introduce a side-channel (in the ideal execution) which might help the adversary to learn a party's secrets by observing if timeouts are triggered or not. In this setting, one could design clearly insecure protocols that *e.g.* realize the ideal commitment functionality $\mathcal{F}_{\text{COM}}$ and prove their security in TLUC.

To this end, consider the following example which illustrate that this is necessary in order to achieve a meaningful notion of security. If environments were to count the actual number of steps depending on the input, this would introduce a side-channel that could leak the secret of honest parties even in the ideal execution. As a consequence, obviously insecure protocols could be proven secure.

As an example, we consider a commitment scheme in the plain model that is clearly insecure but can be shown to realize $\mathcal{F}_{\text{COM}}$ if legal environments count the actually performed steps instead of an upper bound independent of a party's input.

The protocol π between a committer and a receiver is defined as follows:

- Upon receiving $(\texttt{commit}, \mathit{sid}, b)$, the committer sends a message $(\texttt{init}, \mathit{sid})$ to the receiver. The receiver then samples $r_0, r_1 \xleftarrow{\$} \{0,1\}^\kappa$ and sends $(\texttt{timer}, r_0, 10000\kappa)$ to the adversary. Upon its next activation, the receiver sends $(\texttt{timer}, r_1, 1000\kappa)$ to the adversary. Upon its next activation, it sends $(\texttt{init}, \mathit{sid})$ to the committer. If $b = 0$, the committer performs $1000\kappa + 1$ steps. If $b = 1$, the committer performs $10000\kappa + 1$ steps. Afterwards, it sends $(\texttt{done}, \mathit{sid})$ to the receiver. The receiver checks the status of both timers, remembers the result and outputs $(\texttt{committed}, \mathit{sid})$.

- Upon receiving $(\texttt{unveil}, \mathit{sid})$ as input, the committer sends (sid, b) to the receiver. If $b = 0$ and the first timer has timed out but the second has not, it outputs $(\texttt{unveil}, \mathit{sid}, 0)$. If $b = 1$ and both timers have timed out, it outputs $(\texttt{unveil}, \mathit{sid}, 1)$. Otherwise, it halts.

Let $\mathcal{S}$ be the following simulator for π and the dummy adversary.

- If the committer is corrupted, internally run the protocol of the honest receiver. After having received $(\texttt{done}, \mathit{sid})$, interact with $\mathcal{F}_{\text{COM}}$ as follows: If only the first timer has timed out, send $(\texttt{commit}, \mathit{sid}, 0)$ to $\mathcal{F}_{\text{COM}}$ on behalf of the corrupted sender. Otherwise, send $(\texttt{commit}, \mathit{sid}, 1)$. Upon receiving the bit b' during the unveil phase, do the following checks: If $b' = 0$ and only the first timer has timed out, send $(\texttt{unveil}, \mathit{sid})$ to $\mathcal{F}_{\text{COM}}$ on behalf of the corrupted committer and allow the receiver's output. If $b' = 1$ and both timers have timed out, send $(\texttt{unveil}, \mathit{sid})$ to $\mathcal{F}_{\text{COM}}$ on behalf of the corrupted committer and allow the receiver's output. Otherwise, halt.

- If the receiver is corrupted, internally run the protocol of the honest committer on input $b = 0$ after getting the request for the receiver's output from $\mathcal{F}_{\text{COM}}$. When receiving the output request $(\texttt{unveil}, \mathit{sid}, b)$ from $\mathcal{F}_{\text{COM}}$, send (sid, b) to the corrupted receiver.

- If both parties are honest, follow the strategy for the corrupted receiver and additionally simulate the messages of the honest receiver. Eventually, allow the output of $\mathcal{F}_{\text{COM}}$ if the honest receiver would accept.

Now, consider that we require environments to always count the steps relative to the actual steps performed by the protocol parties. Clearly, the protocol is non-trivial (*cf.* Section 3.4.3).

If the sender is corrupted, any legal environment $\mathcal{Z}$ will handle the timers as in the real execution. Thus, the receiver's output is identically distributed as the simulator $\mathcal{S}$ can extract the bit that will be accepted by the receiver (if it exists). Conversely, if the receiver is corrupted, the environment will also count the steps relative to the real execution and, in particular, in dependence of the honest committer's secret and consistent with the value sent by the simulator. Again, the environment's view is identically distributed. The case when both parties are honest is similar.

It follows that any legal environment's view in an execution with π and $\mathcal{D}$ is identically distributed to an execution with IDEAL($\mathcal{F}_{\text{COM}}$) and $\mathcal{S}$ if the steps counted by $\mathcal{Z}$ depend on secret inputs.

Remark A.2. At first glance, the above protocol seems to contradict the impossibility result of Theorem 3.4 since only one party sets up a timer. However, in the presence of "timing side-channels", the ideal execution of $\mathcal{F}_{\text{COM}}$ may not provide meaningful security for the committer. Thus, the setting is different from the one considered in Theorem 3.4, making it inapplicable for this discussion.

Before defining legal environments, we look ahead to the proofs of various properties of our notion. In the proof of the completeness of the dummy adversary (Proposition 3.3), an environments $\mathcal{Z}_{\mathcal{D}}$ interacting with a protocol π and the dummy adversary $\mathcal{D}$ internally runs an adversary $\mathcal{A}$ and a legal environment $\mathcal{Z}$ and appropriately routes all messages. As $\mathcal{Z}_{\mathcal{D}}$ does not perform any meaningful computations on its own but only routes messages, we call it a *routing environment*. In the proof, we have to show that $\mathcal{Z}_{\mathcal{D}}$ is a legal environment. Clearly, the number of steps performed in the interaction of $\mathcal{Z}_{\mathcal{D}}$ with $\mathcal{D}$ is greater than the number of steps in the execution of $\mathcal{Z}$ with $\mathcal{A}$. As $\mathcal{Z}$ is oblivious of the fact that it is emulated, it cannot possibly account for this difference. In particular, in the "outer" execution with $\mathcal{Z}_{\mathcal{D}}$, additional steps are performed by $\mathcal{Z}_{\mathcal{D}}$ due to emulation overhead as well as due to the additional steps performed by the dummy adversary $\mathcal{D}$. Still, we want both executions to count the same number of computation steps. Otherwise, the view of the "inner" environment $\mathcal{Z}$ would not be identically distributed when emulated or $\mathcal{Z}_{\mathcal{D}}$ would not be a legal environment.

In the proof of the single-instance composition theorem (Theorem 3.3), a similar situation arises where an environment $\mathcal{Z}'$ internally emulates an environment $\mathcal{Z}$ and a protocol ρ.

In the following, we thus define the class of routing environments, which are allowed to count steps identically to their internally emulated environment (see Definition A.5), even though more steps are actually performed. This is justified as in every execution with a routing environment, there exists a corresponding "unrolled" execution without routing environments (see Definition A.4) for which the number of counted steps is correct *when this "unrolled" execution is emulated by a single machine.* This unrolled execution can then be used *e.g.* in a reduction, resulting in the same distinguishing advantage, already accounting for the emulation overhead of the reduction adversary.

Non-Routing Environments. We start with the definition of the base case, *i.e.* legal *non-routing* environments. Informally, a non-routing environment $\mathcal{Z}$ is treated "as-is" without accounting for possible other environments, protocols or adversaries internally emulated by $\mathcal{Z}$. We also define what constitutes a legal non-routing environment.

Definition A.2 (Legal Non-Routing Environment). $\mathcal{Z}[\pi, \mathcal{D}]$ is a *legal (type-0) non-routing environment* if it expects to interact with challenge protocol π and adversary $\mathcal{A}$ and, upon receiving ($\texttt{notify}, \mu, id$) from the adversary, immediately sends

- $(\texttt{notify}, \mu, id, 0)$ to the adversary if the party with extended identity μ is honest and the (presumptive) execution of the UC execution experiment with π and $\mathcal{A}$ will have performed less than ℓ steps since
  - $\mathcal{Z}$ has received $(\texttt{timer}, \mu, id, \ell)$ from the adversary and
  - $\mathcal{A}$ has written $(\texttt{notify}, \mu, id, \star)$ to the backdoor tape of μ,

  where the steps are counted as if the whole execution experiment were emulated by a single Turing machine[1]. The calculation of all steps is performed obliviously of the protocol parties' inputs' values, randomness and relative to the first message $(\texttt{timer}, \mu, id, \star)$ from μ with ID *id*.
- $(\texttt{notify}, \mu, id, 1)$ to the adversary otherwise.

Remark A.3. Definition A.2 is somewhat imprecise with respect to the influence of inputs to the calculation of the performed steps. Intuitively, we do not want this computation to depend on an input's actual value *that is not known to the adversary*, *e.g.* a committer's private input, as the number of steps performed by the committer could be different depending on the input.

At the same time, we (usually) do not want the computation to be independent *of the very fact* that an input has (not) been given. In many natural protocols, this information is available to the adversary, *e.g.* from the communication, and thus does (in contrast to the input's value) not need to be protected.

We stress that, at least for protocols with an *a priori* bounded number of inputs (for all parties), the notion is meaningful also if the calculation is performed such that is independent of the given number of outputs, *i.e.* by always assuming that the maximum number has been given. If, however, reactive protocols with an *unbounded* number of possible inputs or protocols with an unbounded number of parties are considered, determining an upper bound of steps possibly performed independent of the number of inputs given may not be possible.

We leave the final choice of how steps should be calculated to the protocol designer, based on the desired security guarantees.

Remark A.4. While we specify that computation steps have to be counted as if the whole TLUC experiment were executed on a single Turing machine, we do not fix the concrete machine model (*e.g.* the alphabet or the number of tapes). The actual machine model we are interested in depends on the timing assumption used in the protocol π (if there are any) such that the environment $\mathcal{Z}$ always counts the correct number of steps if some reduction adversary $\mathcal{A}'$ internally emulates the TLUC execution experiment. As established by Canetti [C01], a Turing machine emulating a UC execution where all entities are PPT is PPT, too.

Routing Environments. With the base case at hand, we can define routing environments that internally emulate routing or non-routing environments as well as an

[1]It is possible to admit a negligible error to these calculation to get a definition that captures more "intuitively secure" protocols.

adversary or a protocol. As the definition of *legal* routing environments is more involved, we defer this to a separate definition.

Definition A.3 (Type-1 and Type-2 Routing Environments)**.** $\mathcal{Z}'[\pi, \mathcal{D}]$ is a *type-1 (routing) environment* if it expects to interact with challenge protocol π and the dummy adversary $\mathcal{D}$ and internally emulates a type-0 legal environment $\mathcal{Z}[\pi, \mathcal{A}]$ and adversary $\mathcal{A}$ that is not the dummy adversary, and routes messages as follows:

- Outputs from the challenge protocol to the environment are forwarded as outputs to the internal environment $\mathcal{Z}$.
- Inputs from the internal environment $\mathcal{Z}$ to the challenge protocol are forwarded to the challenge protocol as inputs of $\mathcal{Z}'$.
- Inputs from the internal environment $\mathcal{Z}$ to the adversary are forwarded to the internal emulation of $\mathcal{A}$.
- Messages from the internal adversary $\mathcal{A}$ to the environment are forwarded to the internal environment $\mathcal{Z}$.
- Messages from the internal adversary $\mathcal{A}$ to the challenge protocol are forwarded to the external adversary as coming from $\mathcal{Z}'$.
- Messages from the adversary to the environment are forwarded to the internal adversary $\mathcal{A}$ as coming from the challenge protocol π.
- Eventually, $\mathcal{Z}'$ outputs what $\mathcal{Z}$ outputs.

$\mathcal{Z}'[\pi, \mathcal{D}]$ is a *type-2 routing environment* if it expects to interact with a challenge protocol π and the dummy adversary $\mathcal{D}$, internally emulates an environment $\mathcal{Z}[\rho^{\phi \to \pi}, \mathcal{D}]$ that is either a non-routing environment, a type-1 routing environment or a type-2 routing environment and protocol ρ that makes one subroutine call to ϕ and routes messages as follows:

- Outputs from the challenge protocol to the environment are forwarded as output to the appropriate party of ρ coming from π.
- Inputs from the protocol ρ to π are forwarded to the challenge protocol as input coming from $\mathcal{Z}'$.
- Inputs from $\mathcal{Z}$ to the adversary pertaining ρ are forwarded to ρ as coming from the dummy adversary $\mathcal{D}$.
- Inputs from $\mathcal{Z}$ to the adversary pertaining π are forwarded to the adversary as coming from the environment $\mathcal{Z}'$
- Messages from ρ to the adversary are forwarded as input to $\mathcal{Z}$ as coming from the dummy adversary $\mathcal{D}$.

- Messages from the adversary to the environment are forwarded to $\mathcal{Z}$.
- Eventually, $\mathcal{Z}'$ output what $\mathcal{Z}$ outputs.

We will use type-1 (legal) environments in the proof of the completeness of the dummy adversary (Proposition 3.3). Type-2 (legal) environments will be used in the proof of the single-instance composition theorem (Theorem 3.3).

As TLUC inherits the notion of polynomial time of the UC framework, routing environments are inherently polynomially bounded. Thus, the nesting depth of routing environments is also polynomially bounded. Also, there always exists an innermost non-routing environment. As routing environments only route messages between the outside and internally emulated entities, there exists an "unrolled" execution without routing environments. In the unrolled execution, the innermost environment interacts with the actual challenge protocol (which may have been split into several parts hosted by different routing environments before) and an adversary which may not be the dummy adversary. We formally define the unrolled execution as follows:

Definition A.4 (Unrolled Execution)**.** Let $\mathcal{Z}, \mathcal{Z}'$ be environments, let ρ, π, ϕ be PPT protocols and let $\mathcal{A}$ be a PPT adversary. We define the unrolled execution of $\mathcal{Z}'$ as follows:

- If $\mathcal{Z}'$ is a (type-0) non-routing environment, the unrolled execution is the execution with environment $\mathcal{Z}'[\pi, \mathcal{A}]$, protocol π and adversary $\mathcal{A}$.
- If $\mathcal{Z}[\pi, \mathcal{A}]$ is a (type-0) non-routing environment emulated by a type-1 routing environment $\mathcal{Z}'[\pi, \mathcal{D}]$, then the unrolled execution of $\mathcal{Z}'[\pi, \mathcal{D}]$ is the execution with environment $\mathcal{Z}[\pi, \mathcal{A}]$, protocol π and adversary $\mathcal{A}$.
- If $\mathcal{Z}[\rho^{\phi \to \pi}, \mathcal{A}]$ (possibly with $\mathcal{A} = \mathcal{D}$) is a (type-0) non-routing environment (transitively) emulated by a type-2 routing environment $\mathcal{Z}'[\pi, \mathcal{D}]$ that externally interacts with a protocol π or ϕ, then the unrolled execution of $\mathcal{Z}'[\pi, \mathcal{D}]$ is the execution with environment $\mathcal{Z}[\rho^{\phi \to \pi}, \mathcal{A}]$, protocol $\rho^{\phi \to \pi}$ resp. ρ and adversary $\mathcal{A}$.

We are now ready to define the class of *legal routing environments*, namely routing environments that correctly handle timers set up by honest parties.

Definition A.5 (Legal Routing Environment)**.** A routing environment $\mathcal{Z}$ expecting to interact with protocol π and adversary $\mathcal{A}$, denoted by $\mathcal{Z}[\pi, \mathcal{A}]$, is called *legal* if upon receiving $(\texttt{notify}, \mu, id)$ from the adversary, it immediately sends

- $(\texttt{notify}, \mu, id, 0)$ to the adversary if the party with extended identity μ is honest and the (presumptive) execution of the UC execution experiment with π and $\mathcal{A}$ will have performed less than ℓ steps since
  - $\mathcal{Z}$ has received $(\texttt{timer}, \mu, id, \ell)$ from the adversary and
  - $\mathcal{A}$ has written $(\texttt{notify}, \mu, id, \star)$ to the backdoor tape of μ,

where the steps are counted as if the whole unrolled experiment (*cf.* Definition A.4) were emulated by a single Turing machine. The calculation of all steps is performed obliviously of the protocol parties' inputs' values, randomness and relative to the first message $(\texttt{timer}, \mu, id, \star)$ from μ with ID id.

- $(\texttt{notify}, \mu, id, 1)$ to the adversary otherwise.

As a consequence of Definitions A.2 to A.5, we establish the following proposition.

Proposition A.1 (Properties of Legal Routing Environments)**.** *Let $\mathcal{Z}$ be the outermost legal non-routing environment that is (transitively) emulated by a legal routing environment $\mathcal{Z}'$. Then,*

- *the number of steps counted by $\mathcal{Z}$ is correct for the unrolled execution run on a single Turing machine and*
- *the view of $\mathcal{Z}$ in the unrolled execution is identically distributed as when (transitively) emulated by $\mathcal{Z}'$.*

Proposition A.1 directly follows from the definition of (legal) routing environments and the definition of the unrolled execution.

Let π be a protocol where no party sends `timer` or `notify` messages to the adversary. Then, every UC environment and every UC adversary is legal for π.

Proposition A.2. *Let $\mathcal{Z}$ be a probabilistic polynomial-time (PPT) environment and let $\mathcal{A}$ be a PPT adversary. Let π be a PPT protocol such that no (sub-)party of π sends* `timer` *or* `notify` *messages to the adversary. Then, $\mathcal{A}$ is a legal adversary and $\mathcal{Z}$ is a legal environment.*

A.2. Analysis of the Committed-Value Oracle $\mathcal{O}_{\mathsf{CCA}}$

In this section, we first recall the robust extraction lemma from [GLP+15] and extend it to our setting. Then, we show that the committed-value oracle $\mathcal{O}_{\mathsf{CCA}}$ from Theorem 4.1 satisfies composition-order invariance.

A.2.1. The Robust Extraction Lemma from [GLP+15]

We recall the rewinding schedule of [GLP+15], which itself is based on [PRS02; PTV14]. In [GLP+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 $(\mathtt{Type}, \text{values})$. Moreover, since we need PRS commitments w.r.t. different base commitments, we make the type of the PRS session explicit. Thus, we have following message types, where m is the "actual" message. Firstly, messages which are irrelevant to the rewinding schedule:

- $(\mathtt{Init}, s, \mathsf{type})$: Initiate PRS session s of type type
- $(\mathtt{Other}, s, m)$: These are all other messages (to and from $\mathcal{A}$) which are not covered below (*e.g.* the commit 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:

- $(\mathtt{Start}, s, m)$: Start of challenge-response in PRS session s.[2] (Sent by $\mathcal{A}$.)
- $(\mathtt{Chall}_i, s, m)$: Challenge for i-th slot of PRS session s. (Sent by PRS oracle.)
- $(\mathtt{Resp}_i, s, m)$: Response to i-th slot of PRS session s. (Sent by $\mathcal{A}$.)
- $(\mathtt{End}, s, m)$: Extracted PRS session result. (Sent by PRS oracle.)
- $(\mathtt{ExtSend}_i, m)$: The i-th message from $\mathcal{A}$ to $\mathcal{B}$. (Sent by $\mathcal{A}$.)
- $(\mathtt{ExtResp}_i, m)$: The i-th response from $\mathcal{B}$ to $\mathcal{A}$. (Sent by PRS oracle.)

Remark A.5. The usual PRS preamble definition and analysis does not consider commitment schemes with a CRS as setup. However, such setups do not affect the rewinding schedule in any way, since they occur before the $(\mathtt{Start}, s, \mathsf{type})$ message (for session s) which start the cut-and-choose phase—the rewinding schedule and extraction analysis is only concerned with the challenge-response slots. Indeed, for our purposes, the CRS is communicated to the adversary by the PRS oracle anyway.

Remark A.6 (Simplifying Assumptions)**.** We assume for simplicity that $\mathcal{A}$ sends the first message to $\mathcal{B}$, and that $\mathcal{B}$ always responds. Moreover, we consider w.l.o.g. *well-behaved* adversaries $\mathcal{A}$, which never send a message which is not expected. They never send a message for a non-existing session, or start the same session multiple times, or skip a response, or continue aborted sessions, *etc.* The PRS receiver could check this and ignore bad messages (or abort session, depending on the PRS preamble behavior), but it simplifies the description to put these validity checks into the adversary.

With notation in place, we recall the rewinding schedule of [GLP$^+$15] (adapted to our notation and typed base commitments). We ignore the messages $(\mathtt{Init}, s, \mathsf{type})$ and $(\mathtt{Other}, s, m)$ 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

[2]Either m is the last message of the commit phase step 1. Or one lets $(\mathtt{Other}, s, m)$ finish that step and sets $m = \bot$ here. Either way, the message is a "start marker" initiating the first challenge. The choice does not affect the rewinding schedule of admissible adversaries.

challenges $(\texttt{Chall}_i, s, m)$ and gather (fresh) responses, while respecting external messages $(\texttt{ExtResp}_i, m)$ which cannot be rewound. We use the PRS preamble as defined in Construction 4. The **procedure** $\mathsf{recurse}(t, \mathsf{st}, \mathcal{T}, \mathsf{f}, \mathrm{aux}, id)$ 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** $\mathsf{recurse}(1, \mathsf{st}, \mathcal{T}, \mathsf{f}, \mathrm{aux}, id)$

1. If the next message is $(\texttt{Start}, s, m, \mathsf{type})$, start a new session s:[3]
   - Let $\mathsf{type}_{(s,id)} = \mathsf{type}$. (Record the session type explicitly.)
   - Send $(\texttt{Chall}_1, s, r_1)$ for $r_1 \leftarrow \mathcal{C}$, where $\mathcal{C} = \{0,1\}^\kappa$.
   - Add $((s, id), 1, r_1, m)$ to $\mathcal{T}$.
2. If the next message is $(\texttt{Resp}_i, s, m)$:
   - 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 $\mathcal{T}$. Otherwise, m is a "good" response and continue.
   - If $i \in \{1, \dots, \ell\}$
     - Add (s, i, r_i, m) to $\mathcal{T}$
     - If $i < \ell$: Send $(\texttt{Chall}_i, s, r_i)$ for $r_i \leftarrow \mathcal{C}$.
     - If $i = \ell$: Send $(\texttt{End}, s, \mathsf{extract}(s, id, \mathcal{T}, \mathsf{aux}))$.
3. If the next message is $(\texttt{ExtSend}_i, m)$:
   - If $\mathsf{f} = 0$, *i.e.* this is a *look-ahead* thread, **return** $(\mathsf{st}, \mathcal{T})$. (Early return.)
   - If $\mathsf{f} = 1$, *i.e.* this is the *main thread*, then:
     - For every live session $s \in \mathsf{LIVE}(\mathsf{st})$ do:[4]
       - Set $\times_{s,id'} = 1$ for every block id' that contains id, including $id' = id$.
     - Send m to $\mathcal{B}$ and receive response m'. Forward $(\texttt{ExtResp}_i, m')$ to $\mathcal{A}$.
4. If not early returned, update the state st to be the current state of $\mathcal{A}$ and return $(\mathsf{st}, \mathcal{T})$.

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.

[3]W.l.o.g., we assume that s uniquely identifies a session *among all threads*. This can be achieved by using (s, id) instead of s as the unique handle for a session. (For any id', there is a unique id, such that id and id' lie on the same thread and $((s, id), 1, \cdot, \cdot) \in \mathcal{T}$; namely, id is the elementary block where session s started (on this thread). This disambiguates identifier s to (s, id) in all situations.)

[4]$\mathsf{LIVE}(\mathsf{st})$ denotes all initiated sessions s which are alive (*i.e.* not completed or aborted) at atomic block st.

Recursive Case: **procedure** $\mathsf{recurse}(t, \mathsf{st}, \mathcal{T}, \mathsf{f}, \mathrm{aux}, id)$

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

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

1. Search in $\mathcal{T}$ for a pair of $((s, id'), i, r_i, m)$, $((s, id''), i, r_i', m')$ 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 $\times_{s,id_1} = 1$.
   - Let id_1' be the sibling[5] of id_1 with input/output tables $\mathcal{T}_{\mathsf{in}}$, $\mathcal{T}_{\mathsf{out}}$ respectively.
   - Attempt to extract (as before) from $\mathsf{aux}_{id_1'} := \mathcal{T}_{\mathsf{in}} \setminus \mathcal{T}_{\mathsf{out}}$.
   - If all attempts fail, return `ExtFail`, otherwise return the extracted value.

Remark A.7 (Ambiguous Extraction)**.** In $\mathsf{extract}(s, id, \mathcal{T}, \mathrm{aux})$, 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 A.8 ((Hierarchically) Structured Randomness)**.** The procedure $\mathsf{recurse}$ is deterministic in all recursive calls, except the base calls. It will be helpful to assume that $\mathsf{recurse}$ interprets its randomness in a structured manner into disjoint/independent parts as follows:

- A tuple $(r_{id}^{\mathtt{Chall}})_{id}$ 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 $(r_{id}^{\mathtt{Other}})_{id}$ which specifies randomness for all other probabilistic computations, *e.g.* the receiver randomness used in base commitments in the PRS commit phase.

[5]The sibling of a block/identity if the other block/identity in the paired calls, *e.g.* the sibling of $id \,\|\, 1$ is $id \,\|\, 2$ and vice versa.

In particular, it is possible to *identify and fix the randomness of the main thread*, and thus all messages and challenges "sent" by $\mathsf{recurse}$ on the main thread. This separation of randomness into atomic blocks will be conceptually helpful later.

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

Lemma A.1 (Robust Concurrent Extraction, adapted from [GLP+15])**.** *Let COM' be the base commitment used in the PRS commitment and suppose that COM' has a stateless receiver. Let ℓ be the number of rounds of the PRS preamble. Let $\mathcal{E}$ be a black-box extractor with extraction based on the rewinding schedule $\mathsf{recurse}$ with extraction method $\mathsf{extract}$. 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 $(\mathtt{ExtSend}_i, m)$ messages of $\mathcal{A}$.*

1. ***Extraction failure.*** *Let $\mathrm{E}_{\mathtt{ExtFail}}$ be the event that in the execution $\mathcal{E}^{\mathcal{A}}$, the extraction returned $\mathtt{ExtFail}$. Then*
$$\Pr[\mathrm{E}_{\mathtt{ExtFail}}] \leq 2^{-\ell+(k+2)\log(M)} + M^2/|\mathcal{C}|$$
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 $M(\kappa)^2\mathsf{poly}(\kappa)$ where $\mathsf{poly}(\kappa)$ is the worst-case runtime 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 unveil phase (and $v \neq \bot$), where v_s is the extracted value. Then*
$$\Pr[\mathrm{B}] \leq \frac{1}{M \cdot \ell \cdot 2\kappa} \,\mathrm{Adv}^{\mathrm{Binding}}_{\mathcal{B},\mathsf{COM}'}(\kappa)$$
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 [GLP+15, Lemma 1]. The efficiency can be derived from $\mathsf{recurse}$ directly (and is also part of [GLP+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.[6] Recall

[6]In Lemma A.4, 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 [GLP+15], stateless receivers are not explicitly required for the validity constraint. See Remark A.13 for a discussion.

that the base commitment scheme has non-interactive decommitments 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.

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. $\square$

A.2.2. Enhanced Robust Extraction Lemma

In this section, we consider an adversary which has access to a *deterministic stateless* complexity oracle $\mathcal{O}_{\mathsf{comp}}$ which allows to break assumptions. Let $\mathcal{H}_{\mathcal{L}}$ denote the bookkeeping algorithm for $\mathcal{O}_{\mathsf{comp}}$, *i.e.* the algorithm which keeps track of invalidated algorithms. We consider w.l.o.g. *admissible* adversaries, which only query $\mathcal{O}_{\mathsf{comp}}$ for oracles $\mathcal{O}_{\mathrm{name}}$ which they have previously enabled by sending $(\mathtt{invalidate}, \mathrm{name})$ to $\mathcal{H}_{\mathcal{L}}$ (analogous to Definition 4.15). More concretely:

- When receiving $(\mathtt{invalidate}, \mathrm{name})$, the bookkeeping algorithm $\mathcal{H}_{\mathcal{L}}$ adds name to $\mathcal{L}$, thereby invalidating assumption name. Initially, $\mathcal{L}$ is empty.
- The adversary $\mathcal{A}$ may (legally) query the complexity oracle $\mathcal{O}_{\mathsf{comp}}$ with $(\mathtt{oracle}, \mathrm{name}, m)$ only if name $\in \mathcal{L}$, *i.e.* if name was previously invalidated. *Admissible* adversaries obey this restriction.

In our security games, we always composed the pseudo-oracle $\mathcal{O}_{\mathsf{CCA}}$ with $\mathcal{A}^{\mathcal{O}_{\mathsf{comp}}}$ (for admissible adversaries). That is, the power of $\mathcal{O}_{\mathsf{comp}}$ is encapsulated within $\mathcal{A}^{\mathcal{O}_{\mathsf{comp}}}$. This requires us to rewind $\mathcal{O}_{\mathsf{comp}}$, which is in general not desirable. Fortunately, since by assumption $\mathcal{O}_{\mathsf{comp}}$ is *deterministic* and *stateless*, rewinding $\mathcal{O}_{\mathsf{comp}}$ does not lead to any artifacts, since it does not affect the behavior of $\mathcal{O}_{\mathsf{comp}}$ at all. We discuss possible generalizations in Appendix A.2.3.

The following is the statement of enhanced robust concurrent extraction, where changes w.r.t. Lemma A.1 are highlighted.

Lemma A.2 (Enhanced Robust Concurrent Extraction)**.** *Let $\mathcal{O}_{\mathsf{comp}}$ be a deterministic stateless complexity oracle. Let COM' be the base commitment used in the PRS commitment and suppose that COM' has a stateless receiver. Let ℓ be the number of rounds of the PRS preamble. Let $\mathcal{E}$ be a black-box extractor with extraction based on the rewinding schedule* $\mathsf{recurse}$ *with extraction method* $\mathsf{extract}$*. 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 $(\mathtt{ExtSend}_i, m)$ messages of $\mathcal{A}$, and let k_{ass} be the maximal number of assumptions which $\mathcal{A}$ invalidates.*

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

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

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. Consequently, if S bounds the worst-case runtime of $\mathcal{A}^{\mathcal{O}_{\mathsf{comp}}}$ plus the PRS receiver and bookkeeping overhead per message. then $\mathcal{E}$ runs in time roughly M^2S.*

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 unveil phase, where v_s is the extracted value, and $v \neq \bot$, and the base commitment is not trivially broken w.r.t. $\mathcal{O}_{\mathsf{comp}}$ (*i.e. $\mathsf{insecure}_{\mathsf{type}(s)}(\mathcal{L})$*). Then*

$$\Pr[\mathrm{B}] \leq \frac{1}{M \cdot \ell \cdot 2\kappa} \operatorname{Adv}^{\text{enh-Binding}}_{\mathcal{B},\mathsf{COM}',\mathcal{O}_{\mathsf{comp}}}(\kappa)$$

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

Proof. This follows completely analogous to Lemma A.1. Indeed, one simply considers the parallel composition $\mathcal{H}_{\mathcal{L}} \parallel \mathcal{B}$ as the left side, and observes that:

- $\mathcal{A}^{\mathcal{O}_{\mathsf{comp}}}$ sends at most $k_{\mathsf{ass}} + k$ messages to the left side.
- If B occurs, then the pair (v, v_s) breaks the enhanced binding property of COM' in the analogous reduction (because the type $\mathsf{type}(s)$ of the commitment is not broken by definition of B).

□

A.2.3. Oracles and Rewinding: Design Decisions

We explain why we will consider *deterministic stateless* (complexity) oracles only, how this relates to a different definition of extractors, and how we could handle probabilistic stateless oracles. We take a look at different ways to handle (complexity) oracles for the rewinding-based extraction procedure.

Oracles and Rewinding. In our rewinding-based extraction, we only consider an algorithm $\mathcal{B}$. Whether $\mathcal{B}$ is a machine or $\mathcal{B} = \mathcal{A}^{\mathcal{O}}$ is an oracle machine with access to $\mathcal{O}$ makes no difference. However, in the latter case, rewinding $\mathcal{B}$, intuitively, requires rewinding $\mathcal{A}$ and $\mathcal{O}$, so if we *implemented* black-box rewinding access to $\mathcal{B}$ *given only* black-box rewinding access to $\mathcal{A}$ and *oracle access* to $\mathcal{O}$, then we would be in big trouble if $\mathcal{O}$ were stateful. Unfortunately, this is the most natural interpretation: While formally, there is no problem in providing black-box rewinding access to $\mathcal{O}$, morally, this is of very different quality compared to black-box access to $\mathcal{A}$. The entity $\mathcal{A}$ is an ordinary machine whose code we should know. But, a priori, an oracle is something extraordinary and should always be seen as external—it is meant to encapsulate some special power after all. Thus, there is a huge difference between *ordinary oracle access* to $\mathcal{O}$ and *black-box rewinding access* to $\mathcal{O}$, which gives an oracle $\mathcal{O}' = \mathsf{bbrw}(\mathcal{O})$ far more powerful than $\mathcal{O}$. Indeed, following the "empowered" extractor definition is a natural alternative to our choice.

Empowered Extractors. Let $\mathcal{O}$ be any deterministic *stateless* oracle $\mathcal{O}$. In the above, we considered $\mathcal{A}^{\mathcal{O}}$ as the adversary and the extractor $\mathcal{E}$ received access to $\mathcal{A}^{\mathcal{O}}$ only. That is, $\mathcal{E}$ was *denied* direct access to the oracle $\mathcal{O}$. We could alternatively consider an extractor $\mathcal{E}'$ which is given access to $\mathcal{O}$ and black-box (rewinding) access to $\mathcal{A}$, and must implement queries to $\mathcal{O}$ for $\mathcal{A}$. Since $\mathcal{O}$ is stateless and deterministic, it is trivial for $\mathcal{E}'$ to implement $\mathcal{O}$ for $\mathcal{A}$. We stress that the notion using $\mathcal{E}'$ strictly empowers the extractor (hence weakens the security), since in principle, $\mathcal{E}'$ could use $\mathcal{O}$ with queries which $\mathcal{A}$ would never use, perhaps trivializing the task of extraction. Thus, this additional power must be treated with care and is best avoided.

We stress that the "empowered" extractor $\mathcal{E}'$ is only more powerful than $\mathcal{E}$ because we assume a (deterministic) *stateless* oracle $\mathcal{O}$. If the oracles were stateful, $\mathcal{E}'$ might not be able to emulate $\mathcal{E}$, because black-box rewinding access to $\mathcal{A}^{\mathcal{O}}$ cannot be emulated with black-box rewinding access to $\mathcal{A}$ and ordinary oracle access to $\mathcal{O}$. Then, the powers of $\mathcal{E}'$ and $\mathcal{E}$ are incomparable.

Probabilistic Oracles. By our definition of black-box (rewinding) access to $\mathcal{B}$, the random tape of $\mathcal{B}$ is chosen uniformly and then fixed. (In particular, the random tape of $\mathcal{B}$ is not provided to or given by the calling party.) This allows to handle *stateless* probabilistic oracles $\mathcal{O}$ basically the same as deterministic ones. Indeed, we already argued that we can w.l.o.g. consider only deterministic adversaries $\mathcal{B}$ for extraction. The same applies to $\mathcal{B} = \mathcal{A}^{\mathcal{O}}$ when $\mathcal{O}$ is probabilistic.

Remark A.9. The setting where the oracle $\mathcal{O}$ uses fresh random coins in every call is not immediately captured by *stateless* $\mathcal{O}$. Due to statelessness, identical queries lead to the same results. However, this is easily resolved by adding a nonce (and partitioning the random tape or having a random oracle baked into $\mathcal{O}$) to provide a mechanism to separate all queries. Any algorithm which expects access to (stateful) $\mathcal{O}$ which uses fresh randomness in every call is easily adapted to this setting (*e.g.* by using a counter as nonce).

A.2.4. Composition-Order Invariance of $\mathcal{O}_{\mathsf{CCA}}$

In this section, we prove the k-robust composition-order invariance of $\mathcal{O}_{\mathsf{CCA}}$ from Theorem 4.1. For this, we consider an adversary $\mathcal{A}$ with access to $\mathcal{O}_{\mathsf{CCA}}$ and an external protocol $\mathcal{B}$, so that the interaction $\langle \mathcal{B}, \mathcal{A} \rangle$ has at most k rounds. Again, we start with the simpler variant without complexity oracle, and then outline the changes required to handle the complexity oracle.

Remark A.10. 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 $\mathsf{recurse}$ anymore. For example, in a system composed of three machines and a pseudo-oracle $\mathcal{O}_{\mathsf{CCA}}$, we can compose the system in several ways:

- $\langle \mathcal{C}, \langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\mathsf{CCA}}} \rangle \rangle$: Here, all messages from $\mathcal{A}$ and to $\mathcal{B}$ or $\mathcal{C}$ are external (for $\mathcal{O}_{\mathsf{CCA}}$), whereas $\mathcal{C}$ and $\mathcal{B}$ are the single external entity to $\mathcal{O}_{\mathsf{CCA}}$. Indeed, this is equivalent to $\langle \mathcal{C} \parallel \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\mathsf{CCA}}} \rangle$.

- $\langle \mathcal{C}, \langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}_{\mathsf{CCA}}} \rangle$: 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 $\mathcal{O}_{\mathsf{CCA}}$).

- $\langle \mathcal{B}, \langle \mathcal{C}, \mathcal{A} \rangle^{\mathcal{O}_{\mathsf{CCA}}} \rangle$: Same as above, with roles of $\mathcal{B}$ and $\mathcal{C}$ swapped.

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

We restate the definitions of COM and $\mathcal{O}_{\mathsf{CCA}}$ from Theorem 4.1. Recall that the construction of COM from typed base commitment COM' (with at most k rounds) with round parameter $\ell \in \omega(k(\kappa)\log(\kappa))$ is as follows:

- **Inputs:** Common input is $(1^\kappa, t)$. Private input to C is v.
- **Setup:** Setup for PRS_ℓ (*i.e.* for base commitment COM'_t and ℓ rounds).
- **Commit Phase:**
  1. **PRS commit:** Run the (typed) PRS commit phase of PRS_ℓ for type t. Let τ_{prs} be the PRS commitment transcript.
  2. **Argument of Knowledge (AoK):** Run Blum's graph hamiltonicity AoK protocol κ-fold in parallel with base commitments COM'_t to prove: τ_{prs} is a valid PRS commitment to some value $v \in M$.
- **Unveil Phase:** Run the corresponding PRS unveil phase.

The oracle $\mathcal{O}_{\mathsf{CCA}}$ is defined as:

- $\mathcal{O}_{\mathsf{CCA}}$ allows $\mathcal{A}$ to choose common inputs $(1^\kappa, t)$ and interact with an honest receiver R_s in session s in arbitrarily many concurrent sessions. For this, $\mathcal{O}_{\mathsf{CCA}}$ first generates a fresh setup $\mathrm{ck}_s \leftarrow \mathsf{Setup}(1^\kappa, t)$ (per type and session) and sends it to $\mathcal{A}$.

- $\mathcal{O}_{\mathsf{CCA}}$ runs the rewinding-based extraction of PRS commitments as in [GLP+15], *cf.* Appendix A.2.1 for more details. Let v_s denote the extracted value (which may be $\bot$) received in (main thread) session s. (If extraction failed, v_s is the special symbol $\bot_{ext}$.)

- When the commit phase of session s completes, $\mathcal{O}_{\mathsf{CCA}}$ outputs $(\mathtt{End}, s, v_s, \mathit{view}_{\mathsf{R}_s})$ where v_s is replaced by
  - $\bot$ if R rejected (the AoK), or
  - $\mathtt{broken}$ if $\mathsf{insecure}_{\mathsf{type}(s)}(\mathcal{L}) = 1$, where $\mathcal{L}$ is the list of invalidated assumptions.[7]

Note that $\mathcal{O}_{\mathsf{CCA}}$ generates the setup and outputs the receiver's view $\mathit{view}_{\mathsf{R}_s}$, unlike the PRS extractor. This does not affect security in any way. Indeed, setup generation is clearly not a problem. And outputting the receiver's view is also trivial, since in Theorem 4.1, it is assumed that receivers are *stateless* anyway.

The Basic Lemma

We first prove composition-order invariance in the standard setting, *i.e.* without a complexity oracle.

Lemma A.3. *Let $\mathcal{A}$, $\mathcal{B}$, $\mathcal{O}_{\mathsf{CCA}}$ as above and recall 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 $\mathcal{A}$ sends 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*

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

Then, there exists an adversary $\mathcal{A}_{\mathsf{COM}'}$ against the binding property of COM' with expected[8] *runtime bounded roughly by the (strict) runtime of extractor $\mathcal{E}^{\langle \mathcal{B}, \mathcal{A}\rangle}$ (*cf. *Lemma A.1). Concretely, if $\langle \mathcal{B}, \mathcal{A}\rangle$ has worst-case runtime S, then $\mathcal{E}^{\langle \mathcal{B}, \mathcal{A}\rangle}$ and $\mathcal{A}_{\mathsf{COM}'}$ have expected runtime bounded roughly by $2 \cdot M^2 S$. In particular, if $\mathcal{B}$ and $\mathcal{A}$ are PPT, then $\mathcal{A}_{\mathsf{COM}'}$ is expected polynomial time. For $\mathcal{A}_{\mathsf{COM}'}$, it holds that for every $\kappa \in \mathbb{N}$ and every $z \in \{0,1\}^*$:*

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

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

The proof idea is straightforward: Whenever $\mathsf{extract}$ 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* $\Pi_1 = \langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\mathsf{CCA}}}\rangle$ and $\Pi_2 = \langle \mathcal{B}, \mathcal{A}\rangle^{\mathcal{O}_{\mathsf{CCA}}}$. Indeed, running the extractor,

[7] Note that any (black-box) $\mathcal{O}_{\mathsf{CCA}}$ can trivially reconstruct $\mathcal{L}$, *i.e.* by recomputing all (invalidation) messages $\mathcal{A}$ sent.

[8] Expected runtime 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.

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 $\mathcal{O}_{\mathsf{CCA}}$, *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, $\mathcal{O}_{\mathsf{CCA}}$ (*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 A.8. 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 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 $\mathcal{O}_{\mathsf{CCA}}$. Note that we assume (Remark A.8) that the randomness r is of the form $r = (r_{id})_{id \in \{1,2,3,4\}^{\log(M)}}$ such that all atomic blocks use disjoint randomness. W.l.o.g., the rewinding sets $t = M$. Let $\Pi_1 = \langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\mathsf{CCA}}} \rangle$ and $\Pi_2 = \langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}_{\mathsf{CCA}}}$. Since the extractor $\mathcal{E}$, *i.e.* recurse, is used to implement the PRS extraction in $\mathcal{O}_{\mathsf{CCA}}$, 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 $\mathcal{O}_{\mathsf{CCA}}$ may send messages which are not identical in Π_1 and Π_2. We say that (the responses of $\mathcal{O}_{\mathsf{CCA}}$ 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 lockstep 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 $\mathrm{F}_1 \vee \mathrm{F}_2$ 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 $\mathrm{F}_{\neq}$.

By Lemma A.1 and a union bound, the probability of an extraction failure for the run with Π_1 or Π_2 is at most

$$\Pr[\mathrm{F}_1 \vee \mathrm{F}_2] \leq 2 \cdot (2^{-\ell + (k+2)\log(M)} + M^2/|\mathcal{C}|).$$

In the following, we consider *modified* outputs out_1, out_2 which always output 0 if $\mathrm{F}_1 \vee \mathrm{F}_2$ occurred. (For this, they run both Π_1 and Π_2 with the same randomness.) The change in statistical distance is at most $\Pr[\mathrm{F}_1 \vee \mathrm{F}_2]$. Thus, from now on, we can ignore the failure case $\mathrm{F}_1 \vee \mathrm{F}_2$.

Next, we show following claim:

Claim A.1. $\Pr[\mathrm{F}_{\neq}] \leq 2^{-\kappa} + \frac{1}{\mathsf{poly} \cdot T} \cdot \mathrm{Adv}^{\mathrm{Binding}}_{\mathcal{A}_{\mathsf{COM}'},\mathsf{COM}'}(\kappa)$.

The lemma then immediately follows. To prove Claim A.1, first denote by s^* the first session (on the main thread) where the divergence of Π_1 and Π_2 occurs. Consider the experiment G where after the occurrence of $\mathrm{F}_{\neq}$, 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 $j \in \{1, \ldots, \ell\}$. Note already here that, even though the extraction of the AoK uses rewinding, it will never rewind *before* the PRS commit phase step 1 ended (on the main thread).

Observe that $\mathrm{F}_{\neq}$ 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 $2^{-\kappa}$.[9] Call this event $\mathrm{F}_{\mathsf{coll}}$.
  - 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 $\mathsf{recurse}$ extracted for Π_1 or Π_2. Again, this yields a binding break.

We let experiment G output 1 if one of the above happens, and 0 otherwise. Then, by definition,

$$\Pr[\mathrm{F}_{\neq}] \leq \Pr[\mathrm{out}_G = 1].$$

Now, we construct an adversary $\mathcal{A}_{\mathsf{COM}'}$ against the binding property of COM', which essentially runs G. Note that $\mathcal{A}_{\mathsf{COM}'}$ will have to rewind $\mathcal{A}$, but cannot rewind the embedded binding game, so this requires some care. Unsurprisingly, $\mathcal{A}_{\mathsf{COM}'}$ runs Π_1 and Π_2 in lockstep, also simulating $\mathcal{B}$, and then 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, $\mathcal{A}_{\mathsf{COM}'}$ must also play the receiver in look-ahead threads, where it cannot embed the binding game anymore. That is, $\mathcal{A}_{\mathsf{COM}'}$ 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 receivers*. Thus, $\mathcal{A}_{\mathsf{COM}'}$ can simply continue the execution of any COM' receiver.

[9] 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.

Observe, that since the randomness in all atomic blocks is independent (*cf.* Remark A.8), 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 $\mathrm{F}_{\neq}$ is unchanged. In full, $\mathcal{A}_{\mathsf{COM}'}$ 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 $i^* \overset{\$}{\leftarrow} \{1, \dots, \mathsf{poly}(\kappa)\}$, where $\mathsf{poly}(\kappa) = \mathsf{poly}_{\mathsf{AoK}}(\kappa) + \kappa \cdot \ell(\kappa)$ is an upper bound for the number of COM' commitments made in a PRS commit 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 $\mathrm{F}_{\neq}$ occurs before session s^*, output $\perp$ to the challenger.[10]
- 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:
  - The rewinding-based AoK extraction occurs strictly *after* the embedded challenge commitment completed, hence $\mathcal{A}_{\mathsf{COM}'}$ never attempts to rewind the challenger.
  - The reduction $\mathcal{A}_{\mathsf{COM}'}$ never provides $(\mathtt{End}, s^*, v_{s^*}, \mathit{view}_{\mathsf{R}_{s^*}})$ to $\mathcal{A}$. Indeed, it could not provide $\mathit{view}_{\mathsf{R}_{s^*}}$, since in general, $\mathit{view}_{\mathsf{R}_{s^*}}$ is only known to the binding challenger.

Overall, this yields our adversary $\mathcal{A}_{\mathsf{COM}'}$ against the 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

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

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

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

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

[10]If $\mathrm{F}_1 \vee \mathrm{F}_2$ occurs before $\mathrm{F}_{\neq}$, the modified experiment immediately outputs 0, so $\mathrm{F}_{\neq}$ will not occur and this case is irrelevant.

Lastly, observe that if $\mathcal{B}$ and $\mathcal{A}$ are PPT, then Π_1 and Π_2 are (overall) PPT algorithms, since $\mathcal{O}_{\mathsf{CCA}}$ is k-robust quasi-PPT. Since $\mathcal{A}_{\mathsf{COM}'}$ essentially runs Π_1 and Π_2 (with minor modifications for embedding its challenge), it is clear that $\mathcal{A}_{\mathsf{COM}'}$ is PPT until the point the AoK extraction starts. (Indeed, $\mathcal{A}_{\mathsf{COM}'}$ makes at most about M^2S steps up to this point, by Lemma A.1.) By the usual argument, in expectation only 1 rewind happens for AoK extraction. (If p is the probability that an AoK challenge is answered acceptingly, then p^{-1} is the expected number of rewinds to obtain another accepting transcript, hence $(1-p)\cdot 0 + p \cdot p^{-1} = 1$ rewinds in expectation.) Thus, in expectation, $\mathcal{A}_{\mathsf{COM}'}$ makes at most about $2M^2S$ steps, as claimed. $\square$

Somewhat tighter reductions are possible by exploiting additional properties of the commitment scheme and the protocol. Most importantly, public-coin receivers can be exploited to embed challenge commitments in every main thread commitment, thus exploiting multi-challenge binding.

The Enhanced Lemma

Now, we consider the enhanced version of Lemma A.3. We define *admissible* adversaries as in Appendix A.2.2. For simplicity, we restate most of the lemma and highlight the changes.

Lemma A.4. *Let $\mathcal{O}_{\mathsf{comp}}$ be a deterministic stateless complexity oracle and let $\mathcal{A}$ be admissible (w.r.t. $\mathcal{O}_{\mathsf{comp}}$). Suppose that $\mathcal{A}$ invalidates at most k_{ass} assumptions. Let $\mathcal{B}$, $\mathcal{O}_{\mathsf{CCA}}$ as in Lemma A.3 and recall 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*

- *$out_1(\kappa, x, y, z)$ as $out_{\mathcal{B},\mathcal{A}}\langle \mathcal{H}_{\mathcal{L}} \parallel \mathcal{B}(x), (\mathcal{A}^{\mathcal{O}_{\mathsf{comp}}})^{\mathcal{O}_{\mathsf{CCA}}}(y)\rangle(1^\kappa, z)$, and*
- *$out_2(\kappa, x, y, z)$ as $out_{\mathcal{B},\mathcal{A}}\langle \mathcal{H}_{\mathcal{L}} \parallel \mathcal{B}(x), \mathcal{A}^{\mathcal{O}_{\mathsf{comp}}}(y)\rangle^{\mathcal{O}_{\mathsf{CCA}}}(1^\kappa, z)$.*

Then, there exists an adversary $\mathcal{A}_{\mathsf{COM}'}$ against enhanced binding property of COM' w.r.t. $\mathcal{O}_{\mathsf{comp}}$ with expected runtime bounded roughly by the (strict) runtime of extractor $\mathcal{E}^{\langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\mathsf{comp}}} \rangle}$ (cf. Lemma A.1). Concretely, if $\langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\mathsf{comp}}} \rangle$ has worst-case runtime S, then $\mathcal{E}^{\langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\mathsf{comp}}} \rangle}$ and $\mathcal{A}_{\mathsf{COM}'}$ have expected runtime bounded roughly by M^2S. For $\mathcal{A}_{\mathsf{COM}'}$, it holds that for every $\kappa \in \mathbb{N}$ and every $z \in \{0,1\}^*$:

$$\begin{aligned}\Delta(out_1(\kappa,x,y,z), out_2(\kappa,x,y,z)) \le 2 \cdot (2^{-\ell+(k_{\mathsf{ass}}+k+2)\log(M)} + M^2/|\mathcal{C}|) + 2^{-\kappa} \\ + \frac{1}{T \cdot \mathsf{poly}(\kappa)} \cdot \mathrm{Adv}^{\text{enh-Bind}}_{\mathcal{A}_{\mathsf{COM}'},\mathsf{COM}'}(\kappa, z).\end{aligned}$$

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

It is possible to handle probabilistic stateless complexity oracles as noted in Appendix A.2.3.

Proof sketch. We only explain how to adapt the proof of Lemma A.3. Again, we exploit that, w.l.o.g., $\mathcal{A}^{\mathcal{O}_{\mathsf{comp}}}$ is an admissible adversary. Observe that $\mathcal{O}_{\mathsf{CCA}}$ outputs `broken` if a base commitment was insecure, thus, this will never result in divergent main threads for Π_1 and Π_2.

Overall, the proof of Lemma A.3 is only minimally modified. Clearly, the presence of $\mathcal{H}_{\mathcal{L}}$, $\mathcal{O}_{\mathsf{comp}}$, and k_{ass} in several expressions must be added. Moreover, the events $\mathrm{F}_{\neq}$ and B are slightly adapted to deal with broken assumptions, but since `broken` cannot lead to divergence, as noted above, these changes hardly affect the proof.

With this, the same analysis for the AoK (and its extraction success) goes through, except that the reduction is now to the enhanced binding property w.r.t. complexity oracle $\mathcal{O}_{\mathsf{comp}}$. □

A.2.5. 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 $\mathcal{O}_{\mathsf{CCA}}$, the substitution rule of interest allows to move an honest receiver session into the $\mathcal{O}_{\mathsf{CCA}}$ oracle, or a session out of the $\mathcal{O}_{\mathsf{CCA}}$ 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 $\mathcal{O}_{\mathsf{CCA}}$ 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 commit phase) as in $\mathcal{O}_{\mathsf{CCA}}$, or whether it will be ignored (*i.e.* replaced by $\top$).
- W_0 forwards everything to $\mathcal{O}_{\mathsf{CCA}}$.
- W_1 forwards only sessions with $e = 1$ to $\mathcal{O}_{\mathsf{CCA}}$, and runs R for $e = 0$.

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

Lemma A.5. *Let $\mathcal{O}_{\mathsf{comp}}$ be a deterministic stateless complexity oracle and let $\mathcal{A}$ be admissible (w.r.t. $\mathcal{O}_{\mathsf{comp}}$). Suppose that $\mathcal{A}$ invalidates at most k_{ass} assumptions. Let $\mathcal{B}$, $W_0^{\mathcal{O}_{\mathsf{CCA}}}$ and $W_1^{\mathcal{O}_{\mathsf{CCA}}}$ 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*

- *$out_1(\kappa, x, y, z)$ as $out_{\mathcal{B},\mathcal{A}}\langle \mathcal{H}_{\mathcal{L}} \parallel \mathcal{B}(x),\, \mathcal{A}^{W_0^{\mathcal{O}_{\mathsf{CCA}}}}(y)\rangle(1^\kappa, z)$, and*
- *$out_2(\kappa, x, y, z)$ as $out_{\mathcal{B},\mathcal{A}}\langle \mathcal{H}_{\mathcal{L}} \parallel \mathcal{B}(x),\, \mathcal{A}^{W_1^{\mathcal{O}_{\mathsf{CCA}}}}(y)\rangle(1^\kappa, z)$.*

*Then, there exists an adversary $\mathcal{A}_{\mathsf{COM}'}$ against enhanced multi-binding (Remark 4.14) with runtime roughly upper bounded by the maximal runtime of $\langle \mathcal{H}_{\mathcal{L}} \parallel \mathcal{B},\, \mathcal{E}^{\mathcal{A}^{W_b^{\mathcal{O}_{\mathsf{CCA}}}}}\rangle$ (for $b = 0$ or 1) (*cf. *Lemma A.1)* in expectation *(in particular, if W_0, W_1, $\mathcal{B}$, $\mathcal{A}$ are PPT, so is $\mathcal{A}_{\mathsf{COM}'}$ (as an oracle algorithm)) such that for every $\kappa \in \mathbb{N}$ and every $x, y, z \in \{0,1\}^*$, it holds that*

$$\Delta(out_1(\kappa, x, y, z), out_2(\kappa, x, y, z)) \leq 2 \cdot (2^{\ell-(k_{\mathsf{ass}}+k+2)\log(M)} + M^2/|\mathcal{C}|) + 2^{-\kappa} + \mathrm{Adv}^{\text{enh-multi-Binding}}_{\mathcal{A}_{\mathsf{COM}'},\mathsf{COM}'}(\kappa, z).$$

where $\mathsf{poly}(\kappa) = \mathsf{poly}_{\mathsf{AoK}}(\kappa) + \kappa \cdot \ell(\kappa)$ and $\mathsf{poly}_{\mathsf{AoK}}$ is a bound on the number of commitments made during in the AoK step, as in Lemma A.3.

Proof sketch. The argument is similar to k-robust composition-order invariance, Lemma A.3. Again, one fixes the randomness of $\mathcal{B}$ and $\mathcal{A}$. Instead of "matching" the randomness of the main threads of $\mathcal{O}_{\mathsf{CCA}}$ in two different executions, as in Lemma A.3, one (fixes and) "matches" the randomness of W_b and $\mathcal{O}_{\mathsf{CCA}}$. (Recall that the randomness of $\mathcal{O}_{\mathsf{CCA}}$ is can be structured suitably to simplify this matching, *cf.* Remark A.8.) That is,

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

Observe that there is an obvious mapping between r' and $(r'_{\mathcal{O}_{\mathsf{CCA}}}, r'_W)$. Moreover, both specify behavior on the main thread completely *as long as extracted values on the main threads do not diverge*,[11] as in Lemma A.3. Following the proof of Lemma A.3, we get a statistical bound on divergence plus a reduction to the 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

$$2 \cdot (2^{-\ell+(k_{\mathsf{ass}}+k+2)\log(M)} + M^2/|\mathcal{C}|) + 2^{-\kappa} + \mathrm{Adv}^{\text{enh-multi-Binding}}_{\mathcal{A}_{\mathsf{COM}'},\mathsf{COM}'}(\kappa, z)$$

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

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

Remark A.11. We formulated Lemma A.5 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, $\mathcal{O}_{\mathsf{CCA}}$ is essentially unaffected by $\mathcal{B}$. Thus, it may be possible to make Lemma A.5 independent of k. For now, this setting appears to be of little interest.

A.2.6. Asides

Example A.1 (PRS is not necessarily COI)**.** The composition-order invariance of PRS commitments depend on their definition of $\mathsf{extract}$ (which, following [GLP+15], we left open in cases of ambiguities). If $\mathsf{extract}$ 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 commitment 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 $\langle \mathcal{B}, \mathcal{A}^{\mathcal{O}_{\mathsf{CCA}}} \rangle$, all extracted slots yield 1. In case $\langle \mathcal{B}, \mathcal{A} \rangle^{\mathcal{O}_{\mathsf{CCA}}}$, there is a non-negligible probability that the slot with 0 is extracted and returned.

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 $\mathsf{extract}$.

Remark A.12. It is not obvious how far the requirements in Lemmata A.3 and A.4 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 A.13 (Necessity of Special Commitment Schemes)**.** While a stateless receiver of COM' may be traded for other (stronger) notions of binding in Lemma A.3, 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 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 Lemma 6 of [GLP+12] (the full version of [GLP+15]). 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 A.1), can be adapted to this setting. Thus, some non-trivial justification (or the restriction of COM') is required for [GLP⁺12, Lemma 6] as well.

Remark A.14 (Relation to [GLP⁺12, Lemma 6])**.** At first glance, Lemmata A.3 and A.4 might be superfluous as the generalized robust concurrent extraction lemma in [GLP⁺12] (the full version of [GLP⁺15]) could be used instead. However, we ran into some obstacles. Firstly, we failed to justify [GLP⁺12, Lemma 6] for general commitment schemes as noted in Remark A.13. And secondly, there are unfortunate ambiguities in [GLP⁺12], so it is not clear if and how their generalized robust concurrent extraction lemma would apply. More precisely:

- The extractor $\mathcal{E}$ in [GLP⁺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 that [GLP⁺12, Lemma 6] claims in constraint (b) is that *for every statistically hiding preamble*, the extracted value 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 [GLP⁺12, Lemma 6], it is clearly explained before and after [GLP⁺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.[12]

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 Lemmata A.3 and A.4.

[12]In the *non*-generalized robust extraction lemma [GLP⁺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 $\mathsf{REAL}^{\mathcal{A}}_{\mathcal{E},\Pi}$. 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 $\mathcal{S}$.