Karlsruhe Institute of Technology

New Frameworks for Concurrently Composable Multi-Party Computation

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

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

genehmigte

Dissertation

von

Brandon Broadnax

aus Karlsruhe

Tag der mündlichen Prüfung:	16.01.2019
Erster Referent:	Prof. Dr. Jörn Müller-Quade
Zweiter Referent:	Prof. Jesper Buus Nielsen, PhD

Acknowledgements

First and foremost, I would like to thank my advisor Jörn Müller-Quade for his invaluable support. We had many interesting discussions, not only on the subject of this thesis, but on many other topics such as the philosophy of mathematics (especially “platonism vs formalism”). I also want to express my gratitude to Jesper Buus Nielsen for taking his valuable time to co-referee my thesis.

Furthermore, I want to thank my co-authors for the many inspiring discussions: Nico Döttling, Valerie Fetzer, Gunnar Hartung, Matthias Huber, Alexander Koch, Bernhard Löwe, Jeremias Mechler, Tobias Müller, Matthias Nagel, Andy Rupp and Patrik Scheidecker.

Special thanks also to Björn Kaidel and Jeremias Mechler for carefully proofreading parts of this thesis.

I had the pleasure to work with many wonderful colleagues. In particular, I want to thank my officemate Björn Kaidel for introducing me to the wonderful world of folk music and for our frequent burger evenings. I also want to express my sincere thanks to Bernhard Löwe for his unwavering kindness and support. Many thanks also to Dirk Achenbach for his help and guidance when I started working at the institute, to Jeremias Mechler for the many interesting discussions on politics, to Thomas Agrikola for always providing me with cough drops, to Antonio Almeida and Mario Strefler for our (“regular”) after-work jogging sessions and for participating in the “Badische Meile” with me, and to all the others for the many joyful moments during my time at the institute.

Another special thanks to Willi Geiselmann, Holger Hellmuth and Carmen Manietta for keeping the institute running behind the scenes.

Last but not least, I thank my friends and family for their encouragement and unending support. In particular, I want to express my deepest gratitude to my deceased maternal grandfather who taught me quite a lot. I dedicate this thesis to him.

Zusammenfassung

Sichere Mehrparteienberechnung ist ein Teilgebiet der Kryptographie, das sich mit dem folgenden Problem befasst: Eine Gruppe von sich gegenseitig misstrauenden Parteien möchte gemeinsam eine Funktion auf ihren geheimen Eingaben berechnen. Um dieses Problem zu lösen, müssen die involvierten Parteien ein *Protokoll* durchführen, das ihnen erlaubt, die gewünschte Funktion gemeinsam zu berechnen ohne sich gegenseitig vertrauen zu müssen. Eine wichtiger Zweig der sicheren Mehrparteienberechnung ist die *komponierbare Mehrparteienberechnung*. Diese hat zum Ziel, Protokolle zu entwickeln, die auch dann noch sicher bleiben, wenn sie in einem Netzwerk mit beliebigen anderen Protokollen ausgeführt werden. Um solche Protokolle zu konstruieren wurden in der Literatur Sicherheitsmodelle entwickelt, die einen *komponierbaren Sicherheitsbegriff* haben, der es erlaubt, Aussagen über die Sicherheit eines Protokolls zu machen wenn es zusammen mit anderen Protokollen ausgeführt wird.

Die vorliegende Dissertation leistet einen Beitrag auf dem Gebiet der komponierbaren Mehrparteienberechnung. Dabei werden zwei neue Sicherheitsmodelle präsentiert und mehrere neue sichere Protokolle konstruiert. Die Hauptbeiträge dieser Arbeit lassen sich wie folgt zusammenfassen:

Ein neues Sicherheitsmodell für komponierbare Mehrparteienberechnung im Standardmodell

Protokolle, die starke Voraussetzungen an eine vertrauenswürdige Infrastruktur (engl. *trusted setup assumptions*) stellen, wie z.B. eine Schlüsselregistrierungsstelle, die alle geheimen Schlüssel der Parteien kennt, sind sehr verwundbar. Insbesondere könnten alle Sicherheitsgarantien verloren gehen wenn diese Infrastruktur korrumpiert wird. Es ist deshalb erstrebenswert Protokolle zu konstruieren, die nur schwache Voraussetzungen an eine vertrauenswürdige Infrastruktur stellen. Vorzugsweise sollten Protokolle im „Standardmodell" (engl. *plain model*) konstruiert werden, in dem nur authentifizierte Kanäle als vertrauenswürdige Infrastruktur vorausgesetzt werden. In den bisherigen Sicherheitsmodellen mit komponierbaren Sicherheitsbegriff können viele interessante kryptographische Aufgaben aber nicht im Standardmodell realisiert werden oder wurden bisher im Standardmodell nur realisiert basierend auf wenig untersuchten oder relativ starken Annahmen oder durch ineffiziente Protokolle, die eine nicht-konstante Anzahl an Kommunikationsrunden benötigen.

Das erste in dieser Arbeit präsentierte Sicherheitsmodell verbessert die aktuelle Situation. Es liefert einen komponierbaren Sicherheitsbegriff, der es erlaubt, für fast alle kryptographischen Aufgaben Protokolle im Standardmodell zu konstruieren, die nur *konstant* viele Runden benötigen und dabei auf gut untersuchten, relativ schwachen Annahmen basieren (sog. *standard polynomial-time assumptions*). Eine der Konstruktionen, die vorgestellt werden, ist zusätzlich vollständig *black-box*. Dies bedeutet, dass die zugrundeliegenden kryptographischen Primitive nur über ihr Eingabe/Ausgabe-Schnittstelle verwendet werden, und insbesondere keine ineffizienten Techniken angewandt werden, die den Code dieser Primitive benutzen. Kein Protokoll für komponierbare Mehrparteienberechnung im Standardmodell, das bisher in der Literatur konstruiert wurde, erfüllte alle der genannten Eigenschaften.

Ein neues Sicherheitsmodell zur Verwendung einfacher, unhackbarer Hardwaremodule

In den bisherigen Sicherheitsmodellen wird zwischen zwei Korruptionsmodellen unterschieden: dem *statischen* Korruptionsmodell, bei dem bereits vor Protokollbeginn feststeht, welche Parteien korrumpiert sind, und dem *adaptiven* Korruptionsmodell, bei dem Parteien während des gesamten Protokollverlaufs korrumpiert werden können. Statische Korruptionen modellieren beispielsweise eine Gruppe böswilliger Protokollteilnehmer, die sich vor Protokollbeginn verschworen haben. Adaptive Korruptionen modellieren Hacker, die während des Protokollverlaufs in Computer einbrechen. Da physikalische Angriffe zeitaufwendig sind und deshalb typischerweise vor Beginn des Protokollverlaufs initiiert werden müssen, können adaptive Korruptionen als *Fernangriffe* aufgefasst werden (z.B. Senden von Viren). Sowohl das statische als auch das adaptive Korruptionsmodell sind sinnvolle Modelle für viele praktische Szenarien. Beide Korruptionsmodelle ignorieren aber die realistische Möglichkeit, dass Parteien nur zu *bestimmten Zeitpunkten* während des Protokollverlaufs durch einen Fernangriff korrumpiert werden können. Beispielsweise könnte eine Partei *einfache, (durch Fernangriffe) unhackbare Hardwaremodule* verwenden wie z.B. *Datendioden*, um unidirektionale Kanäle zu implementieren, oder *(Air-Gap-)Schalter*, um sich vom Netzwerk zu trennen. Ein Angreifer kann deshalb eine Partei nicht unbedingt während des gesamten Protokollverlaufs durch einen Fernangriff korrumpieren, sondern nur während diese *online* ist, d.h. während sie Nachrichten von der Außenwelt empfangen kann. Folglich können die von solchen einfachen, unhackbaren Hardwaremodulen gebotenen Vorteile nicht adäquat in den bisherigen Sicherheitsmodellen abgebildet werden, da diese den Angreifer in seinen Möglichkeiten, Parteien zu korrumpieren, entweder zu sehr oder zu wenig einschränken.

Das zweite in dieser Arbeit vorgestellte Sicherheitsmodell behandelt dieses Problem. Es liefert einen komponierbaren Sicherheitsbegriff und erlaubt, die Vorteile, die von einfachen, unhackbaren Hardwaremodulen wie den oben genannten geboten werden, adäquat abzubilden. Basierend auf sehr wenigen und sehr einfachen unhackbaren Hardwaremodulen werden für fast alle kryptographischen Aufgaben Protokolle konstruiert, die sehr starke Sicherheitsgarantien gegen Fernangriffe bieten.

Abstract

Secure multi-party computation (MPC) deals with the setting where a group of mutually distrustful parties wishes to jointly compute a function of their private inputs. To solve this problem, the parties must engage in an appropriate *protocol* that allows to jointly evaluate the desired function without the need to trust each other. An important branch of MPC is *concurrently composable MPC* which aims at constructing protocols whose security properties remain valid even when they are running concurrently with arbitrary other protocols. In order to obtain concurrently composable MPC protocols, various security frameworks have been proposed that come with a *composable security notion* which allows to argue about the security of a protocol when it is running alongside other protocols.

The present thesis contributes to the field of concurrently composable MPC by providing two new security frameworks along with new secure MPC protocols. The main contributions are summarized as follows:

A New Framework for Concurrently Composable MPC in the Plain Model

Basing the security of protocols on strong *trusted setup assumptions*, such as a key registration authority who knows the secret keys of all parties, makes them very vulnerable. In particular, all security guarantees may be lost if the trusted setup is corrupted. It is therefore desirable to construct protocols requiring only weak assumptions on the trusted setup. The "minimal" setting in this context is the so-called *plain model* where only authenticated channels are assumed as trusted setup. However, in the previous security frameworks with a composable security notion, many interesting cryptographic tasks cannot be realized in the plain model or have so far only been realized in the plain model based on assumptions that are either not well-studied or relatively strong, or by inefficient protocols requiring a super-constant number of communication rounds.

Our first new framework improves upon this by providing a composable security notion that allows the construction of *constant-round* MPC protocols in the plain model for almost every cryptographic task based on well-studied and relatively weak assumptions (so-called *standard polynomial-time assumptions*). One of the constructions we present is additionally fully *black-box*, i.e. only uses the underlying primitives through their input/output interfaces instead of applying inefficient techniques that use the code of these primitives. No prior concurrently composable MPC protocol in the plain model fulfilled all of the aforementioned properties.

A New Framework for Utilizing Simple Remotely Unhackable Hardware Modules

In the existing security frameworks, one distinguishes between two corruption models: the *static* corruption model where the set of corrupted parties is fixed before the protocol execution begins and the *adaptive* corruption model where parties can fall under adversarial control during the entire protocol execution. Static corruptions model, e.g., a group of protocol participants who conspired before the start of the protocol execution. Adaptive corruptions model hackers

actively breaking into computers during a protocol execution. Since direct physical attacks (i.e. tampering with hardware) are time consuming and therefore typically must be mounted before the start of the protocol execution, adaptive corruptions can be regarded as *remote attacks*, e.g., sending of computer viruses. Static and adaptive corruptions are meaningful notions in many practical settings. However, they both ignore the realistic possibility of parties being remotely hackable only at *certain moments* during the protocol execution. For instance, a party may use *simple remotely unhackable hardware modules* such as *data diodes* to implement unidirectional channels or *air-gap switches* to disconnect itself from the network. A hacker may therefore not be able to corrupt a party via a remote attack during the entire protocol execution but only while the party is *online*, i.e. able to receive messages from the outside world. Hence, the advantages provided by such remotely unhackable hardware modules cannot be adequately captured in the existing security frameworks since they give the adversary either not enough or too much freedom over party corruption.

Our second new framework addresses this problem. It provides a composable security notion and allows to adequately capture the advantages provided by remotely unhackable hardware modules such as the ones mentioned above. Utilizing only very few and very simple remotely unhackable hardware modules, we construct MPC protocols for almost every cryptographic task with very strong security guarantees against remote attacks.

Contents

1 Introduction **1**
1.1 Background and Motivation . . . 1
1.2 Contribution of this Thesis . . . 7
1.3 Other Result . . . 10
1.4 Structure of this Thesis . . . 11

2 Preliminaries **13**
2.1 Notation . . . 13
2.2 Basic Concepts . . . 13
2.3 Cryptographic Primitives . . . 14
2.3.1 One-Way Functions . . . 14
2.3.2 Commitment Schemes . . . 15
2.3.3 Public-Key Encryption Schemes . . . 18
2.3.4 Digital Signatures . . . 20
2.3.5 Message Authentication Codes . . . 21
2.4 The Universal Composability Framework . . . 23
2.4.1 Definition of the Framework . . . 23
2.4.2 Standard Well-formed Ideal Functionalities . . . 27
2.4.3 Some Important Functionalities . . . 27
2.4.4 Basic Properties of the Framework . . . 29
2.4.5 Universal Composition . . . 29

3 Non-malleability and CCA Security **31**
3.1 Notions for Public-Key Encryption Schemes . . . 31
3.1.1 Variants of CCA Security . . . 31
3.1.2 Variants of Non-malleability . . . 33
3.1.3 Relations . . . 34
3.2 Notions for Commitment Schemes . . . 35
3.2.1 Contribution . . . 36
3.2.2 Related Work . . . 38
3.2.3 Variants of CCA Security . . . 38
3.2.4 Variants of Non-malleability . . . 40
3.2.5 First Transformation (Puzzle-Solution Approach) . . . 42
3.2.6 Second Transformation (Sharing Approach) . . . 50

4 A New Framework for Concurrently Composable MPC in the Plain Model 57
4.1 Introduction . . . 57
4.1.1 Contribution . . . 60
4.1.2 Related Work . . . 61
4.2 Definitions of the Previous Frameworks . . . 63
4.2.1 SPS Security Framework . . . 63
4.2.2 Angel-based Security Framework and UC with super-polynomial Helpers Framework . . . 63
4.3 Shielded Oracles . . . 64
4.3.1 Definition of the Framework . . . 64
4.3.2 Basic Properties and Justification . . . 67
4.3.3 Universal Composition . . . 69
4.3.4 Polynomial Simulatability . . . 71
4.3.5 Relation with Angel-based Security . . . 74
4.4 A Constant-Round Commitment Scheme . . . 77
4.4.1 Construction and Security Proof . . . 78
4.4.2 First Application: Constant-Round (Black-Box) ZK . . . 85
4.4.3 A Modular Composition Theorem . . . 86
4.5 Constant-Round (Black-Box) General MPC . . . 94

5 A New Framework for Utilizing Simple Remotely Unhackable Hardware Modules 97
5.1 Introduction . . . 97
5.1.1 Contribution . . . 99
5.1.2 Related Work . . . 100
5.2 The Fortified Universal Composability Framework . . . 101
5.2.1 Channels . . . 101
5.2.2 Online State . . . 102
5.2.3 Corruption Model . . . 104
5.2.4 Interface Modules and Fortified Functionalities . . . 107
5.2.5 Notify Transport Mechanism and Activation Instructions 111
5.2.6 Fortified UC Emulation . . . 112
5.3 Properties of the Framework . . . 113
5.4 Construction for Non-Reactive Functionalities . . . 118
5.4.1 Up to N Parties Under Adversarial Control . . . 135
5.5 Construction for Reactive Functionalities . . . 137
5.5.1 Up to N Parties Under Adversarial Control . . . 142
5.6 Architectures without Erasure . . . 142

6 Conclusion and Prospects 145

Bibliography 147

Own Publications 161

Chapter 1

Introduction

1.1 Background and Motivation

Secure Multi-Party Computation. In the setting of multi-party computation (MPC), a group of mutually distrustful parties wants to jointly evaluate a function on their private inputs. In order to do so, the parties must engage in an appropriate *protocol* that allows them to jointly evaluate the desired function without having to trust each other. Research in the area of MPC started with the work of [Yao82] which introduced *Yao's Millionaires' Problem*. This problem can be stated as follows:

Two millionaires want to know who is richer without revealing any additional information about their wealth.

Multi-party computation allows to solve many interesting, sometimes seemingly impossible tasks. For instance, each of the following problems has been solved in the literature with an appropriate MPC protocol:

Private Set Intersection
: Two agencies each hold a set of private data. They want to determine which secrets are held by both agencies without revealing any other information (see, e.g. [FNP04; Dac+09; HEK12]).

Dining Cryptographers Problem (Boolean OR)
: A group of parties sitting around a table in a restaurant are informed that their meals have been paid for by someone. They then want to find out if their meals have been paid by one of them or by a third party not present at the table. However, no party wants to disclose if it paid for the meal or not (see, e.g. [Cha88; WP+89; GJ04]).

(1-out-of-2) Oblivious Transfer
: One party can choose to learn exactly one out of two bits held by another party, while the latter does not learn which bit was chosen by the former (see, e.g. [Rab81; EGL85; NP01; Lin08]).

An important subfield of MPC is *general multi-party computation* which aims at constructing protocols for realizing (almost) *every* function. [Yao86] constructed the first general MPC protocol for the case of *two* parties by introducing the technique of *garbled circuits*. Their protocol works roughly as follows: One party creates an "encrypted circuit" of the desired function with its input hardwired into the circuit. The other party can learn the decryption keys corresponding to its input using an oblivious transfer protocol and can thus evaluate the circuit. This protocol is only secure in the presence of *semi-honest* adversaries who follow the protocol specification but try to learn additional information by analyzing the messages they receive. [GMW87] later gave a solution for the general case of *malicious* adversaries and for an arbitrary number of parties. They showed how a semi-honest protocol can be *compiled* into a protocol secure against malicious adversaries by requiring the parties to prove in each step (via a zero-knowledge proof) that they have honestly followed the protocol specification. Since then, many more general MPC protocols secure against malicious adversaries have been proposed in the literature, e.g. [BGW88; CGT95; CDM00; Ish+06; DN07; IPS08; Cho+09; Lin13; LOS14; Lin+15].

Beyond the question of mere realizability, several research directions focus on constructing general MPC protocols with *additional properties*. These properties are typically related to efficiency or the strength of the required computational assumptions. In the following, we list some of these properties:

Round Complexity
: The number of *(communication) rounds* required by a protocol is measured as a function of the security parameter. In order to minimize the overhead caused by rounds of communication, it is desirable to construct *constant-round* protocols which require only a number of rounds that is independent of the security parameter.

Strength of Required Computational Assumptions
: The computational assumptions required by MPC protocols are generally divided into two categories: *standard polynomial-time assumptions* that are well-studied in the literature or *non-standard* and *super-polynomial-time assumptions* that are either not well-studied or relatively strong compared to the former.

Use of Underlying Primitives (Black-Box vs. Non-Black-Box)
: A protocol is *black-box* if it only uses the underlying cryptographic primitives through their input/output interfaces. *Non-black-box* constructions, on the other hand, use the code of the underlying primitives. It is typically more desirable to obtain black-box protocols as non-black-box constructions tend to be highly inefficient.

Defining Security of MPC Protocols. Finding a meaningful security definition for MPC protocols is a delicate task. One might be inclined to choose the straightforward approach of specifying a list of requirements for a given task and then deeming a protocol for that task "secure" if it fulfills all the requirements on that list. However, this approach is problematic for several reasons.

First, important requirements may be forgotten. As an example, consider the following list of security requirements:

Input Privacy:
: Each party learns no additional information about the other parties' inputs beyond what can be inferred from its own input and output.

Correctness:
Each party outputs the correct result.

It is tempting to believe that an MPC protocol is secure if it fulfills the above list of requirements. However, this is not true. Consider the following toy protocol for the task of computing the exclusive OR (XOR) of two inputs:

Party A sends its input to party B. B then computes the XOR of A's input and its own input and sends the result to A. Both parties then output the computed value.

This protocol fulfills all the requirements on the above list. In particular, input privacy is not an issue for the task of computing the XOR of two inputs. This is because each party can always learn the other party's input by simply XORing its own input to its output. However, the above protocol allows B to learn A's input *before* it chooses its own input. As a consequence, B can determine the result of the protocol alone. Intuitively, this should not be possible in a secure protocol for XOR, i.e. the parties' inputs should be *independent* of each other. Hence, the above list of security requirements is incomplete.

Second, giving formal definitions of security requirements in the presence of malicious parties is difficult. For instance, consider the above-mentioned requirement of input privacy. Since a malicious party may deviate from the protocol specification in arbitrary way, the messages it sends may not correspond to any valid input value. Therefore, it is a-priori not well-defined what it means that such a malicious party can learn no additional information "beyond what can be inferred from its own input and output".

The modern approach for defining security, which informally originated in the work of [GMW87], is based on the so-called *real-ideal paradigm.* In this approach, a given protocol is compared with an "ideally secure" setting, called *ideal protocol.* The parties in an ideal protocol do not directly communicate with each other but instead secretly hand their inputs to an external trusted party, called *ideal functionality*, who locally computes the desired function and secretly hands the parties their outputs. A protocol π is said to *securely realize* a task if it is "as secure as" the ideal protocol ϕ for that task. This means that any "damage" that can be caused by an adversary $\mathcal{A}$ interacting with π can also be caused by an adversary $\mathcal{S}$, called the *simulator*, interacting with the ideal protocol ϕ. If this holds, π is also said to *emulate* ϕ. Many security notions have been formulated based on this approach (e.g. [MR91; BCG93; Can00; Can01; Gol04]) and numerous protocols have been shown to satisfy such a security notion. For instance, [Gol04] gave a detailed proof that the aforementioned general MPC protocol constructed in [GMW87] fulfills their security notion.

Protocol Composition. In practice, cryptographic protocols typically run in a network with other protocols. It is therefore not sufficient to argue the security of a protocol in the *stand-alone setting*, where it is executed once in total isolation (as is done in the frameworks proposed in, e.g., [MR91; BCG93; Can00; Gol04]). To illustrate this, consider the following two examples.

The first example, taken from [Can06], shows that running multiple executions of a protocol concurrently may bring forth new security requirements that do not exist in the stand-alone setting.

Consider the task of *commitment* which is a two-party task consisting of two phases: a *commit phase*, where a sender is able to provide a receiver a “commitment” to a value while keeping that value secret (*hiding property*) and an *unveil phase*, where the sender can open the commitment only to the value he committed to in the commit phase (*binding property*).

Consider a *first-price sealed-bid auction* protocol where each party commits to its bid using a commitment protocol. Once the bidding is over, all commitments are opened. The actioneer then determines the highest submitted bid and thus the winner of the auction.

At first glance, any secure commitment protocol seems to suffice for the above application. Since the commitment is hiding, no party can learn the bid of another party during the bidding stage. Furthermore, the binding property ensures that no party can increase its bid depending on the opened bids. However, the hiding and binding property do not rule out the following attack: A malicious party M may modify a commitment to a value x generated by another party P in such a way that when P opens its bid then M is able to open his commitment to, say, $x + 1$. This way, M can break the security of the auction protocol without breaking the binding or hiding property. Instead, M breaks a new requirement, namely maintaining “independence” between committed values. This requirement, called *non-malleability* in the literature, does not arise in the stand-alone setting since it is of no concern there. However, it may become crucial in a setting where multiple instances of a protocol are executed concurrently such as the commitment protocol in the above auction protocol.

The second example, taken from [GK90; Fei91; Can06], shows that a protocol may be secure in the stand-alone setting but lose security properties (proven in the stand-alone setting) when run concurrently with other protocols. This may even happen when only two instances of the same protocol are run concurrently.

Consider the task of *zero-knowledge* (ZK) where, given a binary (NP-)relation R, a “prover” P holding a value x and a secret “witness” w wants to convince a “verifier” V that $R(x, w) = 1$ holds without revealing anything beyond the validity of that statement (in particular, without revealing w). This task is formally defined by the ideal protocol ϕ_{ZK} in which the ideal functionality $\mathcal{F}_{\mathsf{ZK}}$ takes (x, w) as input from P and outputs $R(x, w)$ to V.

Assume there exists a “puzzle system” that allows the prover and verifier to generate “puzzles” p with the following properties:

- The prover can solve any puzzle.
- The verifier cannot feasibly solve puzzles. He cannot even distinguish between a valid solution to a puzzle generated by himself and a random (invalid) one.

Such a puzzle system exists if one allows the prover to be computationally unbounded [GK90] or by assuming that the prover holds some “trapdoor” information [Fei91].

Let π be any protocol that emulates ϕ_{ZK} according to a security notion in the stand-alone setting, such as one of the notions formulated in [MR91; BCG93; Can00; Gol04]. (Note that since the following argument is informal the exact definition of security is not important. However, formal proofs can be given in each of the aforemention frameworks.) Construct a new protocol π' out of π as follows: First, the parties run the protocol π. Once π has terminated, they continue as follows: P sends a random puzzle p to V. V then generates a puzzle p' and responds with a tuple (s, p'). If s is a correct solution to the puzzle p generated by P, then P sends its secret witness w to V. Otherwise, P sends a solution s' for the puzzle p' generated by V.

It holds that if π is a secure protocol for ZK in the stand-alone setting then so is π'. Informally, this is because V cannot solve puzzles and is therefore unable to make P reveal its witness in a stand-alone setting. Furthermore, since V cannot distinguish the solution s' from a random value, which V could have generated by itself, V does not learn anything in the interaction with P in a stand-alone execution of π' beyond the validity of the given statement.

However, a malicious V can learn a secret witness when interacting in two concurrent executions of the protocol π' as follows: Let P_1 and P_2 be the two provers interacting with V. V first waits to receive the puzzles p_1 and p_2 from the two provers. V then sends (s, p_2) to P_1 for some arbitrary value s. It then obtains a solution s_2 to p_2 from P_1. V can then send $(s_2, \tilde{p})$ for some arbitrary puzzle $\tilde{p}$ to P_2. Since s_2 is a correct solution for p_2, V will obtain P_2's witness.

The Universal Composability Framework. The foregoing examples show that analyzing the security of a protocol only in the stand-alone setting can lead to devastating effects when the analyzed protocol is executed in a network with other protocols. One should therefore have a security notion that also considers the concurrent setting. Formulated via the real-ideal paradigm, a security notion X should fulfill the following property:

If π emulates ϕ (according to security notion X), then any system of protocols where one or multiple instance(s) of π are executed concurrently with arbitrary other protocols emulates the same system of protocols where each instance of π is replaced by an instance of ϕ.

A security notion with this property is called *closed under protocol composition* or *composable*. Security notions that only consider the stand-alone setting do not have this property. For instance, the second example showed that a system consisting of two concurrent executions of a ZK protocol proven secure in the stand-alone setting may fail to emulate a system consisting of two concurrent executions of the ideal protocol ϕ_{ZK}. This is because a malicious verifier may learn a secret witness in the former setting (as shown in the example) but not in the latter since the two instances of the ideal functionality $\mathcal{F}_{\mathsf{ZK}}$ only output the result of the evaluated relation to the verifier.

The *Universal Composability (UC) framework*, put forward by [Can01], provides a security notion that is closed under protocol composition. It does so by introducing an additional entity called *environment* that models protocols running concurrently with the protocol whose security is being analyzed. The environment may freely interact with the adversary throughout the protocol execution and provide inputs to and receive outputs from the protocol parties. A protocol π is said to emulate a protocol ϕ in the UC framework if for every adversary $\mathcal{A}$ interacting with π, there exists a simulator $\mathcal{S}$ interacting with ϕ such that no environment $\mathcal{Z}$ can distinguish between an interaction with π and $\mathcal{A}$ or ϕ and $\mathcal{S}$. As a consequence of this definition, a protocol proven secure in the UC framework is guaranteed to be *(concurrently) composable*, i.e. its security properties remain valid even when multiple instances of that protocol are concurrently executed in an arbitrary unknown environment. The UC framework has become the “standard” framework for designing and analyzing concurrrently composable multi-party computation protocols. However, it has various shortcomings. Two of these shortcomings are the subject of this work and will be described in the following.

Problem 1: Setup Assumptions. Unfortunately, the strong composable security notion provided by the UC framework comes with a price: It has been shown that many interesting tasks such as oblivious transfer, zero-knowledge or commitments cannot be realized with UC security in the *plain model*, where only authenticated channels are assumed as trusted setup [CF01; CKL03; Lin03; PR08; KL11]. Furthermore, [Lin04] proved that the need for some additional trusted setup extends to even the special case of (concurrent) *self*-composability, where only instances of the *same* protocol are concurrently executed. All general MPC protocols that have been constructed in the UC framework therefore assume some additional trusted setup (see, e.g. [Can+02; Bar+04; Can+07; KLP07; Kat07; CPS07; LPV09; Dac+13a; HV15]). For instance, [Can+02] gave a construction based on a *common reference string* where the trusted setup is assumed to provide a string that is honestly sampled from some predefined distribution. However, protocols relying on some additional trusted setup are less likely to be secure in practice, where a trusted setup is often hard to come by (or expensive). It is therefore desirable to have a meaningful composable security notion that can be achieved in the plain model.

In order to reach this goal, new frameworks have been proposed relaxing the security notion of the UC framework. One of the most prominent solutions is based on the idea of providing the simulator with *super-polynomial resources.* This idea is motivated by the fact that for many cryptographic tasks, such as commitments or oblivious transfer, the respective ideal protocol is information-theoretically secure, i.e. remains (ideally) secure even in the presence of adversaries with unbounded runtime. Granting the simulator super-polynomial resources therefore still leads to a meaningful security notion for many cryptographic tasks. This approach was put forward in [Pas03] which introduced the *Security with super-polynomial simulators* (SPS) framework where the simulator is allowed to run in super-polynomial-time. Many self-composable general MPC protocols in the plain model have been constructed in the SPS security framework (see, e. g., [Pas03; BS05; LPV12; Gar+12; Dac+13a; GKP17; GKP18]). However, a major drawback of the SPS security framework is that its security notion is not closed under protocol composition. In particular, SPS-secure protocols can become insecure when executed alongside other protocols.

In order to overcome this problem of SPS security, [PS04] proposed the *Angel-based security framework.* In this framework, the adversary, environment and simulator run in polynomial-time but all have access to a so-called *Imaginary Angel* that provides super-polynomial resources for *specific* computational problems. The security notion in the Angel-based security framework is closed under protocol composition and implies SPS security. [CLP10] later put forward the *UC with super-polynomial helpers* framework where the environment and simulator may query a super-polynomial *helper* which, unlike Imaginary Angels, may be interactive and stateful. This framework also provides a composable security notion implying SPS security. Many general MPC protocols in the plain model have been constructed in the Angel-based security and UC with super-polynomial helpers framework. However, all of these constructions are either based on non-standard or super-polynomial-time assumptions [PS04; MMY06; KMO14] or require a super-constant number of rounds [CLP10; LP12; Kiy14; Goy+15; HV16].

Problem 2: Corruption Model. In the UC framework one (essentially) distinguishes between two corruption models: the *static* corruption model and the *adaptive* corruption model. The static corruption model considers adversaries that corrupt parties only *before* the protocol execution begins. This captures, e.g., a group of protocol participants who conspired prior to the start of the protocol execution. In contrast, adversaries in the adaptive corruption model (first proposed by [Can+96]) are able to corrupt parties throughout the entire protocol execution, allowing them to base their corruption strategy on the messages they received so far. Intuitively, adaptive corruptions model hackers who actively break into computers during a protocol execution. Since direct physical attacks (i.e. tampering with hardware) are time consuming and therefore must typically be mounted prior to the start of the protocol execution, adaptive corruptions can be regarded as *remote attacks*, e.g., sending of computer viruses. The UC framework with adaptive corruptions ("adaptive UC framework") captures real-life threats more accurately than its analog with static corruptions. However, it ignores the realistic possibility of a party being (temporarily) isolated from the network and therefore not remotely hackable. For instance, a party could utilize *simple remotely unhackable hardware modules* such as *air-gap switches* to disconnect itself from the network or *data diodes* to implement unidirectional channels. An adversary may therefore not be able to corrupt a party via a remote attack at any given moment but only while the party is *online*, i.e. able to receive messages from the outside world. Furthermore, a party may use additional hardware modules such as a simple encryption unit that only implements a specific public-key encryption scheme. Such hardware modules with very limited functionality can be implemented securely as fixed-function circuits and formally verified for correctness. They can therefore be assumed to be resilient against remote attacks. In particular, an adversary can only corrupt such hardware modules if he has direct physical access to them. Using such hardware modules in conjunction with air-gap switches and data diodes allows a party to, e.g., implement secure message transmission without the risk of being remotely hacked while sending (sensitive) data.

It follows that the advantages provided by simple remotely unhackable hardware modules cannot be adequately captured in the adaptive UC framework since this framework gives the adversary too much freedom over party corruption. It is therefore desirable to find a new framework that also provides a composable security notion but allows to capture these advantages.

1.2 Contribution of this Thesis

This thesis provides two new frameworks for concurrently composable MPC, each of which addresses one of the problems stated in the previous section. Our main contributions are summarized as follows:

A New Framework for Concurrently Composable MPC in the Plain Model

> Our first new framework provides a composable security notion that implies SPS security and can be satisfied by constant-round general MPC protocols in the plain model based on only standard polynomial-time assumptions. One of the constructions we present is additionally fully black-box. No prior concurrently composable general MPC protocol in the plain model fulfilled all of the aforementioned properties.

A New Framework for Utilizing Simple Remotely Unhackable Hardware Modules

> Our second new framework provides a composable security notion and allows to adequately capture the advantages provided by remotely unhackable hardware modules. Utilizing only very few and very simple remotely unhackable hardware modules, we construct general MPC protocols with very strong security guarantees against remote attacks based on standard polynomial-time assumptions.

We note that although one could also try to conceive a single new framework addressing both of the aforementioned problems we have chosen to present a separate framework for each problem instead. This is because these problems involve quite different concepts and techniques. Furthermore, presenting two frameworks instead of one greatly simplifies the presentation of this thesis.

In the following, we give a more detailed description of our contributions:

A New Framework for Concurrently Composable MPC in the Plain Model

We present a new framework that is also based on the idea of providing simulators with super-polynomial resources. However, unlike in previous frameworks, simulators in our new framework are only given *restricted access* to the results computed in super-polynomial time. We model this restricted access via stateful oracles that may directly interact with the ideal functionality without the simulator observing the communication. We call these oracles *shielded oracles*.

As with UC security, the security notion of our framework is closed under protocol composition. Furthermore, our security notion can be shown to lie strictly between Angel-based security/UC security with super-polynomial helpers and SPS security and is compatible with UC security in the sense that protocols proven secure in the UC framework are also secure in our new framework.

Restricting access to super-polynomial resources allows us to apply a new proof technique where entities involving super-polynomial resources can be replaced by indistinguishable polynomially bounded entities. As a consequence, our constructions require weaker building blocks than the general MPC protocols in the Angel-based security and UC with super-polynomial helpers framework. In particular, it suffices to use *parallel* CCA-secure commitment schemes as a building block in our constructions instead of “full-fledged” CCA-secure commitment schemes, which were commonly used in the general MPC protocols in the previous frameworks. Unlike CCA-secure commitment schemes, parallel CCA-secure commitment schemes have known constant-round (and black-box) instantiations based on standard polynomial-time assumptions. As a result, we are able to show that constant-round general MPC in the plain model based on standard polynomial-time assumptions can be achieved in our new framework. We present two general MPC protocols with these properties. One of our protocols is also fully black-box. To the best of our knowledge, this construction was the *first* black-box constant-round general MPC protocol satisfying a meaningful composable security notion in the plain model based on standard polynomial-time assumptions.

A New Framework for Utilizing Simple Remotely Unhackable Hardware Modules

We present a new framework that allows to capture the advantages provided by simple remotely unhackable hardware modules. In our new framework, each party has an *online state* (online/offline) that is determined by the type and state of its *channels*, e.g., state of its air-gap switches. Furthermore, the adversary is able to corrupt parties in two different ways, namely by mounting either *physical attacks*, which model physically tampering with (or replacing) hardware, or *online attacks*, which model remote hacking attacks. Contrary to physical attacks, online attacks give the adversary control over a party only if the party is currently online and *not* assumed to be unhackable. We call our new framework *Fortified UC*.

Our framework provides a security notion that is closed under protocol composition and equivalent to adaptive UC security for protocols that do not use any remotely unhackable hardware modules. As a consequence, UC-secure protocols can be used as building blocks in protocols in our new framework.

Utilizing only very few and very simple remotely unhackable hardware modules, we construct general MPC protocols where mounting online attacks does not enable an adversary to learn or modify a party's inputs and outputs unless he gains control over a party via the input port *before* the party has received its (first) input (or gains control over *all* parties). Hence, our protocols protect against all online attacks, except for online attacks via the input port while a party is waiting for its (first) input. To achieve theses strong security guarantees against online attacks, the parties' inputs and outputs are authenticated, masked and shared in our protocols in such a way that an adversary cannot learn or modify them when corrupting a party through an online attack.

It is important to note that the simple remotely unhackable hardware modules used in our general MPC protocols are based on substantially weaker assumptions than the tamper-proof hardware tokens proposed by [Kat07]. In particular, they are not assumed to be physically tamper-proof and can thus not be passed to other (possibly malicious) parties. Therefore, these remotely unhackable hardware modules cannot be used as an additional trusted setup to circumvent the aforementioned impossibility results of the UC framework. Our general MPC protocols therefore rely on additional, well-established setup assumptions (e.g. a common reference string) in conjunction with simple remotely unhackable hardware modules to achieve concurrent composability. Given all the aforementioned assumptions, our general MPC protocols provide the best possible protection against online attacks in a setting where parties cannot be protected while waiting for input.

Additional Result

Relations between Variants of Non-malleability and CCA security for Commitment Schemes

The setting of *non-malleability*, introduced by [DDN91], considers in its basic form an adversary $\mathcal{A}$ interacting in two sessions Π_1, Π_2 of a two-party protocol Π such as a commitment scheme, zero-knowledge protocol or a public-key encryption (PKE) scheme. $\mathcal{A}$ interacts with an honest sender S in session Π_1, playing the role of the receiver. Conversely, $\mathcal{A}$ plays the role of the sender in the other session Π_2, interacting with an honest receiver R. Informally, Π is said to be non-malleable if no adversary can transform a transcript $T(a)$ he receives in session Π_1, which depends on S's secret input a, into a transcript $T'(\tilde{a}) \neq T(a)$ in session Π_2 corresponding to a value $\tilde{a}$ that is *related* to a and accepted by R. For instance, a commitment scheme (informally) satisfies non-malleability if for every message m, no adversay can transform a commitment $\mathrm{Com}(m)$ he receives into another commitment $\mathrm{Com}(\tilde{m})$ to a related message $\tilde{m}$.

Another security notion that is closely related to non-malleability is *CCA security*. Roughly speaking, a two-party protocol Π such as commitment scheme or a PKE scheme is CCA-secure if the secret input of the sender S remains hidden even in the presence of adversaries who have access to an *oracle* $\mathcal{O}$ that extracts input values from transcripts of *other* instances of Π. For instance, a PKE scheme is CCA-secure if (informally) an adversary can learn nothing about a plaintext m from a ciphertext $c \leftarrow \mathrm{Enc}(pk, m)$ (apart from its length) even given an oracle $\mathcal{O}(sk, \cdot)$ that decrypts all ciphertexts $c' \neq c$.

Non-malleable and CCA-secure protocols play a central role as building blocks in concurrently composable general MPC protocols (e.g. [LPV09; CLP10; Gar+12]). This is also true for the general MPC protocols presented in this work which make crucial use of PKE schemes or commitment schemes fulfilling some notion of non-malleability or CCA security. Identifying which notions of non-malleability or CCA security are necessary for our general MPC protocols was an important problem during our research. In particular, we explored how the identified notions relate to other notions in order to assess how strong the identified notions are and, in particular, how they can be instantiated. For PKE schemes, the relations among these notions have already been settled in the literature (e.g. [Bel+98; BS99]). However, only very few relations have been analyzed for commitment schemes so far. We have therefore proved separations between a variety of notions related to non-malleability and CCA security that were proposed for commitment schemes in the literature. These results are presented in this thesis as an additional contribution.

1.3 Other Result

We have also proposed a new approach to software protection that is not part of this thesis. An abstract of this result is given below.

Towards Efficient Software Protection Obeying Kerckhoffs's Principle

Software has become an increasingly important commercial factor. As a consequence, software protection, i.e. protection against *unauthorized copying* of software products, has become an important field of IT security. In order to be effective, software protection inherently requires some form of hardware since software can always be copied if it is not physically protected. However, since hardware is costly, a practical software protection scheme cannot solely rely on hardware. Instead, a combination of practically viable hardware measures and software methods should be employed.

Like any cryptographic scheme, software protection schemes should fulfill *Kerckhoffs's principle* which dictates that a cryptographic scheme should be secure even if everything about it, except for cryptographic keys, is publicly known. A scheme which violates this principle gives an adversary who knows the underlying mechanisms sufficiently well a great advantage, often resulting in the scheme being completely broken. Unfortunately, software protection schemes used in practice generally rely on "security through obscurity" and therefore violate Kerckhoffs's principle. In the literature, various methods have been proposed that can be used to guarantee provably secure software protection in accordance with Kerckhoffs's principle (e.g, [Gol87; GO96; PR10; Shi+11; Lin16; LT17]). However, these constructions are impractical due to their overhead.

In [Bro+18b] we proposed a new software protection scheme that is both compliant with Kerckhoffs's principle and practically viable. At the core of this scheme lies a simple assumption: a hacker lacks the domain knowledge that is necessary to create the software product he seeks to copy illegally. For instance, a hacker may not be familiar with the underlying mathematics of a scientific program such as a computer algebra system.

This lack of domain knowledge can be exploited to obtain a secure protection. The main idea is to *partition* the code of a software in such a way that, given an arbitrary subset of parts, it is hard to come up with the remaining parts if one lacks the underlying domain knowledge. Each part of the partitioned code is encrypted separately and only decrypted when needed during the program execution. The key for decrypting these parts is stored on a hardware dongle.

Assuming a program admits a partition with the above property, a hacker can perform no attack on our scheme that is better than storing every part of the program that is decrypted during program execution. Of course, not all programs admit such a partition. Identifying which code structures admit such a partition is part of ongoing research.

1.4 Structure of this Thesis

Each chapter contains a summary of the contribution presented in that chapter as well as a discussion of the related work. In the following, we give a brief outline of this thesis.

- In Chapter 2, we introduce notation and give definitions of some of the cryptographic primitives used in this work along with various well-known security notions for these primitives. We also give a brief and simplified introducton into the UC framework, which will serve as the basis for the new security frameworks presented in this work.

- In Chapter 3, we introduce several notions related to non-malleability and CCA security that were proposed for PKE schemes and commitment schemes in the literature. The first part of this chapter cites some proven relations among these notions for PKE schemes and contains no original work of the author. The second part clarifies the relations among these notions for commitment schemes. This part is taken almost entirely from [Bro+18a], which was published at PKC 2018. Both parts of this chapter serve as a complement to the following chapters and may be skipped on first reading.

- In Chapter 4, we present a new framework for concurrently composable MPC in the plain model. We show that the security notion of our framework lies stricly between SPS and Angel-based security and is closed under protocol composition. We present two constant-round general MPC protocols in the plain model based on standard polynomial-time assumptions that are secure in our framework. One of these constructions is also fully black-box. This chapter is taken almost entirely from [Bro+17], which was published at EUROCRYPT 2017.

- In Chapter 5, we introduce a new framework that allows to capture the advantages provided by simple remotely unhackable hardware modules. We prove that the security notion of this framework is closed under protocol composition and equivalent to adaptive UC security for protocols that do not use any remotely unhackable hardware modules. Using only very few and very simple remotely unhackable hardware modules, we construct general MPC protocols providing very strong security guarantees against online attacks. This chapter is taken almost entirely from [Bro+18c], which is available as a technical report at the Cryptology ePrint Archive but has not been published elsewhere yet.

- In Chapter 6, we briefly summarize the main contributions of this work and indicate some future research directions related to this work.

Chapter 2

Preliminaries

In this chapter, we introduce notation and give definitions of the cryptographic primitives and security models used in this work.

2.1 Notation

Throughout this work, we denote the *security paramter* by $n \in \mathbb{N}$. We assume that (if not explicitly mentioned) all algorithms, i.e. Turing machines, are implicitly given the security parameter as input in unary form 1^n. We call an algorithm *probabilistic* if it has access to a random tape whose cells each contain a uniformly distributed bit. We say that an algorithm runs in *polynomial time* if its runtime is upper bounded by a polynomial in the bitlength of its input. We call a probabilistic algorithm running in polynomial time a *PPT algorithm*. If an algorithm does not run in polynomial time, then we say that it runs in *super-polynomial time*.

For a PPT algorithm $\mathcal{M}$, we write $y \leftarrow \mathcal{M}(x)$ to denote the output y of $\mathcal{M}$ on input x. We sometimes want to make the randomness used by $\mathcal{M}$ explicit and write $y = \mathcal{M}(x; r)$. We denote by $\mathcal{M}^{\mathcal{O}}(x)$ an algorithm $\mathcal{M}$ with input x and (black-box) access to an oracle $\mathcal{O}$.

For a bitstring $x \in \{0,1\}^*$, we denote by $|x|$ the bitlength of x. For a finite set S, we write $s \leftarrow S$ to denote the operation of picking an element s of S uniformly at random. Furthermore, we write $\text{card}(S)$ to denote the cardinality of S. We denote by $\perp$ a special error symbol and by poly an unspecified polynomial.

2.2 Basic Concepts

In this section, we recall some basic general concepts that will be used throughout this work (cf. [Gol03] for a detailed discussion of these concepts).

Negligible functions are used to formalize the notion of a "very small", and hence tolerable, adversarial success probability. Informally, a function is negligible if its absolute value approaches zero faster than the inverse of any positive polynomial. A formal definition is given below.

Definition 2.1 (Negligible and Overwhelming Function)**.** *A function $f : \mathbb{N} \to \mathbb{R}$ is* negligible *if for every constant $c > 0$ there exists an integer $n_0 \in \mathbb{N}$ such that for all $n > n_0$ it holds that $|f(n)| < n^{-c}$. In the following, we denote by* negl *an unspecified negligible function.*

We call a function $f : \mathbb{N} \to \mathbb{R}$ overwhelming *if $1 - f$ is a negligible function.*

A nice property of negligible functions, which follows easily from the above definition, is that they are closed under addition and multiplication.

Next, we recall the concept of computionally indistinguishable probability ensembles.

Definition 2.2 (Probability Ensembles)**.** *Let $\mathcal{I}$ be a countable set. A* probability ensemble *$X = \{X_i\}_{i \in \mathcal{I}}$ is a sequence of random variables X_i indexed by I.*

Definition 2.3 (Computational Indistinguishability)**.** *Let $X = \{X_{n,z}\}_{n\in\mathbb{N},z\in\{0,1\}^*}$ and $Y = \{Y_{n,z}\}_{n\in\mathbb{N},z\in\{0,1\}^*}$ be probability ensembles. X and Y are* computationally indistinguishable*, denoted by $X \overset{c}{\equiv} Y$, if for every PPT "distinguisher" $\mathcal{D}$ whose running time is polynomial in its first input, there exists a negligible function* negl *such that for all $n \in \mathbb{N}$ and $z \in \{0,1\}^*$ it holds that*[1]

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

2.3 Cryptographic Primitives

In this section, we define cryptographic primitives that are relevant to this work.

2.3.1 One-Way Functions

One-way functions are one of the most important primitives in cryptography. Roughly speaking, a one-way function is easy to compute but hard to invert. The following definition of one-way functions is taken from [Gol03].

Definition 2.4 (One-Way Functions)**.** *A function $f : \{0,1\}^* \to \{0,1\}^*$ is called* one-way *if the following holds:*

(i) There exists a (deterministic) polynomial-time algorithm F such that $F(x) = f(x)$ for all $x \in \{0,1\}^*$.

(ii) For every PPT adversary $\mathcal{A}$, there exists a negligible function negl *such that for all $n \in \mathbb{N}, z \in \{0,1\}^*$ it holds that*

$$\mathop{\mathsf{Pr}}_{x \leftarrow \{0,1\}^n}[\mathcal{A}(1^n, z, f(x)) \in f^{-1}(f(x))] \le \mathsf{negl}(n)$$

(Note that the probability is also over the internal coin tosses of $\mathcal{A}$).

[1]Note that, throughout this work, we only consider security notions in the *non-uniform* setting, i.e. distiguishers/adversaries always have an additional (non-uniform) input. This is because we will work in frameworks based on the UC framework, which is non-uniform.

If f is one-way and additionally a permutation over $\{0,1\}^n$ for every n, then f is called a *one-way permutation* (OWP).

Next, we define trapdoor permutations, which are one-way functions that allow for efficient computation of preimages when given an appropriate trapdoor. This notion was first proposed by [Gol03]. In the following, we give a definition of this primitive, paraphrasing from [Gol04].

Definition 2.5 (Trapdoor Permutations)**.** *A* collection of trapdoor permutations *is a collection of permutations $\{f_\alpha : \mathrm{D}_\alpha \to \mathrm{D}_\alpha\}$ together with four PPT algorithms (I, S, F, B) such that the following holds:*

(i) On input 1^n, algorithm I selects at random an n-bit long index α *(not necessarily uniformly) of a permutation f_α, along with a corresponding* trapdoor τ *and returns (α, τ).*

(ii) On input α, algorithm S samples the domain D_α *of f_α, returning an almost uniformly distributed element in it.*

(iii) For any $x \in \mathrm{D}_\alpha$, given α and x, algorithm F returns $f_\alpha(x)$.

(iv) For any y in the range of $f_\alpha(x)$, if (α, τ) is a possible output of $I(1^n)$, then, given τ and y, algorithm B returns $f_\alpha^{-1}(y)$.

(v) For every PPT adversary $\mathcal{A}$, there exists a negligible function negl *such that for all $n \in \mathbb{N}, z \in \{0,1\}^*$ it holds that*

$$\Pr_{\substack{(\alpha,\tau)\leftarrow I(1^n)\\ y\leftarrow S(\alpha)}}[\mathcal{A}(z, \alpha, y) = f^{-1}(y)] \leq \mathsf{negl}(n)$$

(Note that the probability is also over the internal coin tosses of $\mathcal{A}$).

A stronger primitive than trapdoor permutations are enhanced trapdoor permutations. Enhanced trapdoor permutations fulfill the requirement that it is hard to find the preimage of a random element $y \leftarrow S(\alpha)$ even when given the coins used by S in the generation of y. This primitive was introduced by [Gol04]. In the following, we recall that definition (paraphrasing from [Gol04]).

Definition 2.6 (Enhanced Trapdoor Permutations)**.** *A collection of trapdoor permutations $\{f_\alpha : \mathrm{D}_\alpha \to \mathrm{D}_\alpha\}$ (as defined in Definition 2.5) is called* enhanced *if the following holds: For every PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that for all $n \in \mathbb{N}, z \in \{0,1\}^*$ it holds that*

$$\Pr_{\substack{(\alpha,\tau)\leftarrow I(1^n)\\ r\leftarrow \{0,1\}^{\mathsf{poly(n)}}}}[\mathcal{A}(z, \alpha, r) = f_\alpha^{-1}(S(\alpha; r))] \leq \mathsf{negl}(n)$$

(Note that the probability is also over the internal coin tosses of $\mathcal{A}$).

2.3.2 Commitment Schemes

A *commitment scheme* $\langle C, R\rangle$ is a protocol between two parties, called *sender* and *receiver*, that consists of two phases: a *commit phase* and an *unveil phase.*

In the commit phase, the sender is able to provide the receiver a "commitment" to a value while keeping that value secret (*hiding property*). In the unveil phase, the sender can open the commitment only to the value he committed to in the commit phase (*binding property*). Throughout this work, we require that the binding property is "statistical", i.e., holds against *unbounded* adversaries, while the hiding property is "computational", i.e., only holds against polynomially bounded adversaries. This ensures that a commitment uniquely defines the committed value with overwhelming probability. In addition, we assume a "canonical unveil phase". This means that the sender opens the commitment by sending a tuple (v, r), where r is the randomness used by the sender in the commit phase, and the receiver checks if (v, r) is consistent with the messages he received in the commit phase. Furthemore, we focus on *tag-based* commitment schemes [PR05; DDN00a] in this work. In a tag-based commitment scheme, the sender and receiver additionally use a common "tag" as an additional input to the protocol. Intuitively, a tag is the chosen identitity of the sender of a commitment.

In the following, we formally define the hiding and binding property:

Definition 2.7 (Computationally Hiding). *Let $\mathsf{Exp}^{\mathsf{hiding}}_{\langle C,R\rangle,\mathcal{A}(z)}(n)$ denote the output of the following probabilistic experiment:*

On input $1^n, z$, the adversary $\mathcal{A}$ picks a tag tag and two strings v_0 and v_1 and sends the tuple (v_0, v_1, tag) to the experiment. The experiment then randomly selects a bit $b \leftarrow \{0,1\}$ and commits to v_b by playing the role of the honest sender of $\langle C, R\rangle$ using the tag tag chosen by $\mathcal{A}$. Finally, $\mathcal{A}$ sends a bit b' to the experiment, which outputs 1 if $b = b'$ and 0 otherwise.

An adversary is called valid *if he only chooses strings v_0, v_1 such that $|v_0| = |v_1|$.*

$\langle C, R\rangle$ is said to be computationally hiding *if for every valid PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for all $n \in \mathbb{N}, z \in \{0,1\}^*$ it holds that*

$$\Pr[\,\mathsf{Exp}^{\mathsf{hiding}}_{\langle C,R\rangle,\mathcal{A}(z)}(n) = 1\,] \leq \frac{1}{2} + \mathsf{negl}(n)$$

Definition 2.8 (Statistically and Perfectly Binding). *Let $\mathsf{Exp}^{\mathsf{binding}}_{\langle C,R\rangle,\mathcal{A}(z)}(n)$ denote the output of the following probabilistic experiment:*

On input $1^n, z$, the adversary $\mathcal{A}$ first interacts with the experiment by playing the role of the sender in $\langle C, R\rangle$ using a tag of his choice. After the commit phase is over, $\mathcal{A}$ sends two tuples $(v, r), (v', r')$ to the experiment. The experiment then outputs 1 if $v \neq v'$ and both tuples are accepting, and 0 otherwise.

$\langle C, R\rangle$ is said to be statistically binding *if for every adversary $\mathcal{A}$, there exists a negligible function negl such that for all $n \in \mathbb{N}, z \in \{0,1\}^*$ it holds that*

$$\Pr[\,\mathsf{Exp}^{\mathsf{binding}}_{\langle C,R\rangle,\mathcal{A}(z)}(n) = 1\,] \leq \mathsf{negl}(n)$$

If $\Pr[\,\mathsf{Exp}^{\mathsf{binding}}_{\langle C,R\rangle,\mathcal{A}(z)}(n) = 1\,] = 0$, then $\langle C, R\rangle$ is said to be perfectly binding.

Computationally hiding and statistically binding commitment schemes can be constructed with two rounds of interaction based on any one-way function (see, e.g. [Nao89]). Furthermore, [Blu81] provided a non-interactive commitment scheme that is computationally hiding and perfectly binding based on any one-way permutation.

We will also use homomorphic commitment schemes in this work, which are defined as follows:

Definition 2.9 (Homomorphic Commitment Schemes)**.** *A* homomorphic commitment scheme $\mathsf{Com}_{\mathsf{hom}}$ *consists of two PPT algorithms* (Gen, Commit) *such that:*

1. *The* key generation algorithm Gen *takes as input* 1^n *and outputs a (public)* commitment key ck*. The commitment key specifies three groups, which we write muliplicatively,* $\mathcal{M}_{\text{ck}}$ *(message space),* $\mathcal{R}_{\text{ck}}$ *(randomizer space) and* $\mathcal{C}_{\text{ck}}$ *(commitment space). We assume that one can sample elements uniformly at random from* $\mathcal{R}_{\text{ck}}$*.*
2. *The* commit algorithm Commit *takes as input a commitment key* ck*, a message* $m \in \mathcal{M}_{\text{ck}}$ *and a randomizer* $r \in \mathcal{R}_{\text{ck}}$ *and outputs a commitment* $\text{Commit}(\text{ck}, m, r) \in \mathcal{C}_{\text{ck}}$*.*

In this work, we use verifiable perfectly binding homomorphic commitment schemes*, which are homomorphic commitment schemes with the following properties:*

- (Computationally Hiding) *Let* $\mathsf{Exp}^{\mathsf{hidhom}}_{\mathsf{Com}_{\mathsf{hom}},\mathcal{A}(z)}(n)$ *denote the output of the following probabilistic experiment:*

  At the beginning, the experiment generates $\text{ck} \leftarrow \text{Gen}(1^n)$*. On input* $1^n, z$ *and* ck*, the adversary* $\mathcal{A}$ *picks two messages* m_0, m_1 *and sends the tuple* (m_0, m_1) *to the experiment. The experiment then randomly selects a bit* $b \leftarrow \{0,1\}$ *and computes* $r \leftarrow \mathcal{R}_{\text{ck}}$*,* $\texttt{com}^* = \text{Commit}(\text{ck}, m_b, r)$ *and sends* $\texttt{com}^*$ *to* $\mathcal{A}$*. Finally,* $\mathcal{A}$ *sends a bit* b' *to the experiment, which outputs* 1 *if* $b = b'$ *and* 0 *otherwise.*

  An adversary is called valid *if he only chooses messages* m_0, m_1 *such that* $m_0, m_1 \in \mathcal{M}_{\text{ck}}$*.*

  $\mathsf{Com}_{\mathsf{hom}}$ *is said to be* computationally hiding *if for every valid PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $n \in \mathbb{N}, z \in \{0,1\}^*$ *it holds that*

  $$\mathsf{Pr}[\,\mathsf{Exp}^{\mathsf{hidhom}}_{\mathsf{Com}_{\mathsf{hom}},\mathcal{A}(z)}(n) = 1\,] \leq \frac{1}{2} + \mathsf{negl}(n)$$

- (Perfectly Binding) *For all commitment keys* ck *output by* $\text{Gen}(1^n)$ *it holds that if* $m, m' \in \mathcal{M}_{\text{ck}}$ *such that* $m \neq m'$ *then* $\text{Commit}(\text{ck}, m, r) \neq \text{Commit}(\text{ck}, m', r')$ *for all* $r, r' \in \mathcal{R}_{\text{ck}}$*.*

- (Homomorphism) *For all commitment keys* ck *output by* $\text{Gen}(1^n)$ *and all* $m_1, m_2 \in \mathcal{M}_{\text{ck}}$ *and* $r_1, r_2 \in \mathcal{R}_{\text{ck}}$ *it holds that*

  $$\text{Commit}(\text{ck}, m_1 \cdot m_2, r_1 \cdot r_2) = \text{Commit}(\text{ck}, m_1, r_1) \cdot \text{Commit}(\text{ck}, m_2, r_2)$$

  (where the group operations are carried out in $\mathcal{M}_{\text{ck}}$*,* $\mathcal{R}_{\text{ck}}$ *and* $\mathcal{C}_{\text{ck}}$*, respectively.)*

- (Verifiability) *There exists a (deterministic) polynomial-time algorithm* Verify *such that* $\mathrm{Verify}(\mathrm{ck}) = 1$ *if and only if there exists a string* $r \in \{0,1\}^{\mathsf{poly}(n)}$ *such that* $\mathrm{ck} = \mathrm{Gen}(1^n; r)$.

Verifiable perfectly binding homomorphic commitment schemes can be constructed based on, e.g., the well-studied DDH assumption. Concretely, the ElGamal commitment scheme [ElG84] fulfills these properties (cf. [AIR01]). Moreover, the Linear Encryption scheme [BBS04] can also be viewed as a commitment scheme with these properties. This scheme is based on the DLin assumption, which is a generalization of the DDH assumption.

Finally, we define extractable commitment schemes as proposed by [PW09]. Roughly speaking, an extractable commitment schemes comes with an efficient extraction algorithm which extracts the value committed by a (possibly malicious) sender.

Definition 2.10 (Extractable Commitment Schemes)**.** *Let $\langle C, R\rangle$ be a statistically binding commitment scheme. Then, $\langle C, R\rangle$ is* extractable *if there exists a PPT oracle machine E (the "extractor") such that for any PPT sender S^*, E^{S^*} outputs a pair (τ, σ) such that*

- *τ is identically distributed to the view of S^* at the end of interacting with an honest receiver R in the commit phase.*
- *the probability that τ is accepting and $\sigma \neq \perp$ is negligible.*
- *if $\sigma \neq \perp$, then it is statistically impossible to decommit τ to any value other than σ.*

[PW09] showed how to transform any commitment scheme into a commitment scheme that is also extractable.

2.3.3 Public-Key Encryption Schemes

A public-key encryption scheme allows to encrypt a message using someone's publicly available "public key". Decryption is possible with the corresponding "secret key" that is only known to its owner. In the following, we give a definition of this primitive, paraphrasing from [LK14].

Definition 2.11. *A* public-key encryption scheme PKE *is a tuple of three PPT algorithms* $(\mathrm{Gen}_{\mathrm{PKE}}, \mathrm{Enc}, \mathrm{Dec})$ *such that:*

1. *The* key-generation algorithm $\mathrm{Gen}_{\mathrm{PKE}}$ *takes as input* 1^n *and outputs a pair of keys* $(\mathrm{pk}, \mathrm{sk})$. *We call* pk public key *and* sk secret key.
2. *The* encryption algorithm Enc *takes as input a public key* pk *and a message $m \in \mathcal{M}$ for some underlying message space $\mathcal{M}$, and outputs a ciphertext c.*
3. *The* decryption algorithm Dec *takes as input a secret key* sk *and a ciphertext c and outputs a message $m \in \mathcal{M}$ or a special symbol $\perp$ denoting failure. We assume without loss of generality that* Dec *is deterministic and write* $m = \mathrm{Dec}(\mathrm{sk}, m)$.

It is required that for every $n \in \mathbb{N}$, every $(\mathrm{pk}, \mathrm{sk})$ *output by* $\mathrm{Gen}_{\mathrm{PKE}}(1^n)$ *and every* $m \in \mathcal{M}$ *it holds that*

$$\mathrm{Dec}(\mathrm{sk}, \mathrm{Enc}(\mathrm{pk}, m)) = m$$

PKE schemes should prevent an adversary from learning anything about a plaintext m given a ciphertext $c \leftarrow \mathrm{Enc}(pk, m)$ and the public key pk (apart from its length). This requirement is formally captured by the notion of *indistinguishability under chosen plaintext attacks security* (IND-CPA security), which was put forward by [GM84].

Definition 2.12 (Indistinguishability under Chosen Plaintext Attacks)**.** *Let* $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathrm{PKE},\mathcal{A}(z)}(n)$ *denote the output of the following probabilistic experiment: At the beginning, the experiment generates a key pair* $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{Gen}_{\mathrm{PKE}}(1^n)$*. On input* 1^n*,* z *and* pk*, the adversary* $\mathcal{A}$ *chooses two messages* m_0, m_1 *and sends the tuple* (m_0, m_1) *to the experiment. The experiment then chooses a random bit* $b \leftarrow \{0,1\}$*, computes* $c^* \leftarrow \mathrm{Enc}(\mathrm{pk}, m_b)$ *and sends* c^* *to* $\mathcal{A}$*. At the end of the experiment,* $\mathcal{A}$ *sends a bit* $b' \in \{0,1\}$*. The experiment then outputs* 1 *if* $b = b'$*, and* 0 *otherwise.*

An adversary is called valid *if he only chooses messages* m_0, m_1 *such that* $|m_0| = |m_1|$*.*

A public-key encryption scheme PKE *is* IND-CPA-secure *if for every valid PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $n \in \mathbb{N}, z \in \{0,1\}^*$ *it holds that*

$$\Pr[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathrm{PKE},\mathcal{A}(z)}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$$

It is sometimes convenient to use the following alternative formulation of IND-CPA security:

Equivalent Formulation. Denote by $\mathrm{Output}^{\mathsf{ind\text{-}cpa}}_{\mathrm{PKE},\mathcal{A}(z)}(n, b)$ the output of $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathrm{PKE},\mathcal{A}(z)}(n)$ if the challenge bit is fixed to b.

A public-key encryption scheme PKE is IND-CPA-secure if and only if for every valid PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for all $n \in \mathbb{N}, z \in \{0,1\}^*$ it holds that

$$|\mathrm{Output}^{\mathsf{ind\text{-}cpa}}_{\mathrm{PKE},\mathcal{A}(z)}(n, 0) - \mathrm{Output}^{\mathsf{ind\text{-}cpa}}_{\mathrm{PKE},\mathcal{A}(z)}(n, 1)| \leq \mathsf{negl}(n)$$

IND-CPA-secure PKE schemes can be constructed assuming the existence of trapdoor permutations (cf. [Gol04]).

We will also use PKE schemes with oblivious public key generation as proposed by [DN00]. A PKE scheme with this property allows to sample public keys without obtaining knowledge of the corresponding secret key. In order to formally define this notion, we first give a definition of invertible sampling, which was also put forward by [DN00].

Definition 2.13 (Invertible Sampling)**.** *Let* M *be a PPT algorithm.* M *is said to have* invertible sampling *if there exists a PPT algorithm* M^{-1} *such that* *the probability ensembles* $\{(x, y, r) | r \leftarrow \{0,1\}^{\mathsf{poly}(n)}, y \leftarrow M(x; r)\}_{n \in \mathbb{N}, x \in \{0,1\}^*}$ *and* $\{(x, y, r') | r \leftarrow \{0,1\}^{\mathsf{poly}(n)}, y \leftarrow M(x; r), r' \leftarrow M^{-1}(y, x)\}_{n \in \mathbb{N}, x \in \{0,1\}^*}$ *are computationally indistinguishable.*

We now give a definition of PKE schemes with oblivious public key generation, paraphrasing from [DN00].

Definition 2.14 (PKE Schemes with Oblivious Public Key Generation)**.** *A public-key encryption scheme* $\mathrm{PKE} = (\mathrm{Gen}_{\mathrm{PKE}}, \mathrm{Enc}, \mathrm{Dec})$ *has* oblivious public key generation *if there exists a PPT algorithm* $\widetilde{\mathrm{Gen}}$ *with invertible sampling (as defined in Definition 2.13) such that the probability ensembles* $\{\mathrm{pk} | (\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{Gen}_{\mathrm{PKE}}(1^n)\}_{n \in \mathbb{N}}$ *and* $\{\widetilde{\mathrm{pk}} | \widetilde{\mathrm{pk}} \leftarrow \widetilde{\mathrm{Gen}}(1^n)\}_{n \in \mathbb{N}}$ *are computationally indistinguishable.*

IND-CPA-secure PKE schemes with oblivious public key generation can be constructed based on various well-studied assumptions such as the DDH or RSA assumption (see e.g. [DN00]).

2.3.4 Digital Signatures

A digital signature scheme allows a party to produce a signature of a digital document using its own private "signing key". Each signature can be publicly verified with the "verification key" corresponding to the signing key with which the signature was produced. In the following, we give a definition of digital signature schemes, paraphrasing again from [LK14].

Definition 2.15. *A* digital signature scheme SIG *is a tuple of PPT algorithms* $(\mathrm{Gen}_{\mathrm{SIG}}, \mathrm{Sig}, \mathrm{Vrfy}_{\mathrm{SIG}})$ *such that:*

1. *The* key-generation algorithm $\mathrm{Gen}_{\mathrm{SIG}}$ *takes as input* 1^n *and outputs a pair of keys*$(\mathsf{vk}, \mathsf{sgk})$*. We call* vk *the* verification key *and* sgk *the* signing key*.*

2. *The* signature-generation algorithm Sig *takes as input a signing key* sgk *and a message* $m \in \{0,1\}^*$ *and outputs a* signature σ*.*

3. *The* verification algorithm $\mathrm{Vrfy}_{\mathrm{SIG}}$ *takes as input a verification key* vk*, a message* $m \in \{0,1\}^*$ *and a signature* σ *and outputs a bit* $b \in \{0,1\}$*, with* $b = 1$ *meaning* valid *and* $b = 0$ *meaning* invalid*. We assume without loss of generality that* $\mathrm{Vrfy}_{\mathrm{SIG}}$ *is deterministic and write* $b = \mathrm{Vrfy}_{\mathrm{SIG}}(\mathrm{vk}, m, \sigma)$*.*

It is required that for every $n \in \mathbb{N}$*, every* $(\mathsf{vk}, \mathsf{sgk})$ *output by* $\mathrm{Gen}_{\mathrm{SIG}}(1^n)$ *and every* $m \in \{0,1\}^*$*, it holds that*

$$\mathrm{Vrfy}_{\mathrm{SIG}}(\mathsf{vk}, m, \sigma) = 1$$

In our constructions, we will additionally assume that the digital signature scheme is length-normal, meaning that signatures of messages of equal length are also of equal length.

Definition 2.16 (Length-Normal Signatures)**.** *A digital signature scheme* SIG *is* length-normal *if for every* $n \in \mathbb{N}$*, every* $(\mathsf{vk}, \mathsf{sgk})$ *output by* $\mathrm{Gen}_{\mathrm{SIG}}(1^n)$ *and all* $m, m' \in \{0,1\}^*$ *such that* $|m| = |m'|$ *the following holds: If* $\sigma \leftarrow \mathrm{Sig}(\mathsf{sgk}, m), \sigma' \leftarrow \mathrm{Sig}(\mathsf{sgk}, m')$*, then* $|\sigma| = |\sigma'|$*.*

Digital signature schemes should prevent an adversary from forging signatures. This requirement is formally captured by the notion of *existential unforgeability under adaptive chosen message attacks* (EUF-CMA security). Informally, a digital signature scheme is EUF-CMA-secure if no adversary can produce a forgery even if he his allowed to obtain signatures of any *other* message of his choice. A formal definition of this security notion follows.

Definition 2.17 (Existential Unforgeability under Adaptive Chosen Message Attacks)**.** *Let* $\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathrm{SIG},\mathcal{A}(z)}(n)$ *denote the output of the following probabilistic experiment: At the beginning, the experiment generates a key pair* $(\mathrm{vk}, \mathrm{sgk}) \leftarrow \mathrm{Gen}_{\mathrm{SIG}}(1^n)$*. On input* $1^n, z$ *and* vk*, the adversary* $\mathcal{A}$ *may then send queries to the signing oracle* $\mathcal{O}_{\mathsf{Sig}(\mathrm{sgk},\cdot)}$*. Let* $\mathcal{Q}$ *denote the set of all queries. At the end of the experiment,* $\mathcal{A}$ *outputs a tuple* (m, σ)*. The experiment then checks if* $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathrm{vk}, m, \sigma) = 1$ *and* $m \notin \mathcal{Q}$ *and outputs* 1 *if this holds and* 0 *otherwise.*

A digital signature scheme SIG *is* EUF-CMA-secure *if for every PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $n \in \mathbb{N}, z \in \{0,1\}^*$ *it holds that*

$$\Pr[\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathrm{SIG},\mathcal{A}(z)}(n) = 1] \leq \mathsf{negl}(n)$$

In this work, we will require a weaker notion of unforgeability, namely *existential unforgeability under non-adaptive chosen message attacks* (EUF-naCMA security). Contrary to EUF-CMA security, this notion only considers adversaries who are allowed to obtain signatures of other messages non-adaptively. Below, we give a formal definition of this security notion.

Definition 2.18 (Existential Unforgeability under Non-Adaptive Chosen Message Attacks)**.** *Let* $\mathsf{Exp}^{\mathsf{euf\text{-}nacma}}_{\mathrm{SIG},\mathcal{A}(z)}(n)$ *denote the output of the following probabilistic experiment: At the beginning, the experiment generates a key pair* $(\mathrm{vk}, \mathrm{sgk}) \leftarrow \mathrm{Gen}_{\mathrm{SIG}}(1^n)$*. On input* $1^n, z$*, the adversary* $\mathcal{A}$ *may then send a single parallel query to the signing oracle* $\mathcal{O}_{\mathsf{Sig}(\mathrm{sgk},\cdot)}$*. Let* $\mathcal{Q}$ *denote the set of all queries. Afterwards,* $\mathcal{A}$ *is given the verification key* vk*. Finally,* $\mathcal{A}$ *outputs a tuple* (m, σ)*. The experiment then checks if* $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathrm{vk}, m, \sigma) = 1$ *and* $m \notin \mathcal{Q}$ *and outputs* 1 *if this holds and* 0 *otherwise.*

A digital signature scheme SIG *is* EUF-naCMA-secure *if for every PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $n \in \mathbb{N}, z \in \{0,1\}^*$ *it holds that*

$$\Pr[\mathsf{Exp}^{\mathsf{euf\text{-}nacma}}_{\mathrm{SIG},\mathcal{A}(z)}(n) = 1] \leq \mathsf{negl}(n)$$

Lenght-normal EUF-(na)CMA-secure digital signatures schemes can be constructed assuming the existence of one-way functions (cf. [Rom90] or alternatively [Gol04]).

2.3.5 Message Authentication Codes

Message authentication codes (MACs) are the symmetric analogue to digital signature schemes, i.e. parties have to share a key in order to use this primitive. A definition follows (paraphrasing again from [LK14]).

Definition 2.19. *A* message authentication code MAC *is a tuple of PPT algorithms* $(\mathrm{Gen}_{\mathrm{MAC}}, \mathrm{Mac}, \mathrm{Vrfy}_{\mathrm{MAC}})$*, such that:*

1. *The* key-generation algorithm Gen_{MAC} *takes as input* 1^n *and outputs a key* k. *We call* k *the* MAC key.

2. *The* tag-generation algorithm Mac *takes as input a MAC key* k *and a message* $m \in \{0,1\}^*$ *and outputs a* MAC tag t.

3. *The* verification algorithm Vrfy_{MAC} *takes as input a MAC key* k*, a message* $m \in \{0,1\}^*$*and a MAC tag* t *and outputs a bit* $b \in \{0,1\}$*, with* $b = 1$ *meaning* valid *and* $b = 0$ *meaning* invalid. *We assume without loss of generality that* Vrfy_{MAC} *is deterministic and write* $b = \text{Vrfy}_{\text{MAC}}(k, m, t)$.

It is required that for every $n \in \mathbb{N}$*, every* k *output by* $\text{Gen}_{\text{MAC}}(1^n)$ *and every* $m \in \{0,1\}^*$ *it holds that*

$$\text{Vrfy}_{\text{MAC}}(k, m, t) = 1]$$

As with digital signature schemes, MACs are required to be unforgeable. We will consider two security notions for MACs in this work: the relatively weak notion of *existential unforgeability under one chosen message attacks* (EUF-1-CMA security) and the notion of *existential unforgeability under adaptive chosen message attacks* (EUF-CMA security). The definitions are given below.

Definition 2.20 (Existential Unforgeability under One Chosen Message Attacks). *Let* $\mathsf{Exp}^{\mathsf{euf\text{-}1\text{-}cma}}_{\text{MAC},\mathcal{A}(z)}(n)$ *denote the output of the following probabilistic experiment: At the beginning, the experiment generates a MAC key* $k \leftarrow \text{Gen}_{\text{MAC}}(1^n)$. *On input* $1^n, z$*, the adversary* $\mathcal{A}$ *may send a single query* m' *to an oracle* $\mathcal{O}_{\text{Mac}(k,\cdot)}$. *Afterwards,* $\mathcal{A}$ *outputs a tuple* (m, t). *The experiment then checks if* $\text{Vrfy}_{\text{MAC}}(k, m, t) = 1$ *and* $m \neq m'$ *and outputs* 1 *if this holds and* 0 *otherwise.*

A message authentication code MAC *is* EUF-1CMA-secure *if for every PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $n \in \mathbb{N}, z \in \{0,1\}^*$ *it holds that*

$$\Pr[\mathsf{Exp}^{\mathsf{euf\text{-}1\text{-}cma}}_{\text{MAC},\mathcal{A}(z)}(n) = 1] \leq \mathsf{negl}(n)$$

Definition 2.21 (Existential Unforgeability under Adaptive Chosen Message Attacks). *Let* $\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\text{MAC},\mathcal{A}(z)}(n)$ *denote the output of the following probabilistic experiment: At the beginning, the experiment generates a MAC key* $k \leftarrow \text{Gen}_{\text{MAC}}(1^n)$. *On input* $1^n, z$*, the adversary* $\mathcal{A}$ *may send queries to an oracle* $\mathcal{O}_{\text{Mac}(k,\cdot)}$. *Let* $\mathcal{Q}$ *denote the set of all queries. FAt the end of the experiment,* $\mathcal{A}$ *outputs a tuple* (m, t). *The experiment then checks if* $\text{Vrfy}_{\text{MAC}}(k, m, t) = 1$ *and* $m \notin \mathcal{Q}$ *and outputs* 1 *if this holds and* 0 *otherwise.*

A message authentication code MAC *is* EUF-CMA-secure *if for every PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $n \in \mathbb{N}, z \in \{0,1\}^*$ *it holds that*

$$\Pr[\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\text{MAC},\mathcal{A}(z)}(n) = 1] \leq \mathsf{negl}(n)$$

The works of [GGM84; GGM86] imply that EUF-CMA secure MACs (and hence also EUF-1-CMA-secure MACs) can be constructed given any one-way function (see also [Gol04]). Moreover, [WC81] showed how to construct MACs which satisfy EUF-CMA security even in the presence of unbounded adversaries.

2.4 The Universal Composability Framework

In this section, we give a brief and simplified introduction into the Universal Composability (UC) framework. For a detailed description of the UC framework, see [Can01].

2.4.1 Definition of the Framework

The Basic Model of Computation. In the UC framework, the model of computation is based on interactive Turing machines (ITMs). An ITM is a Turing machine with the following tapes:

- identity tape
- work tape
- random tape
- outgoing message tape
- *Externally writable* tapes (for holding inputs coming from other machines).
  - input tape
  - incoming message tape
  - subroutine output tape
- one-bit activation tape

Instances of an ITM, referred to as ITIs, are identified by the contents of their identity tape. We assume that the identity of an ITI is a tuple (PID, SID) where PID is called the *party identifier* and SID the *session identifier.*

Intuitively, an ITM is an algorithm written for a distributed system (i.e. a "protocol", cf. paragraph "Cryptographic Protocols"), while an ITI is a specific entity in a distributed system running the code of that algorithm. Different entities within a distributed system are differentiated with their PIDs.

External Write Instructions. ITIs are able to communicate with each other via `external write` *instructions* that are written on the sending ITI's outgoing message tape. One distinguishes three different ways in which an ITI can provide information to another ITI: *provide input*, *send a message* or *give subroutine output.* This is modelled by `external write` instructions containing either the targeted ITI's input tape, incoming message tape or subroutine output tape as one of its arguments. In the UC framework, not all `external write` instructions written by an ITI are permitted. Instead, a *control function* decides if an `external write` instruction is allowed or silently dropped (cf. paragraph "UC Execution Experiment" for a description of the allowed `external write` instructions).

Cryptographic Protocols. A *protocol* π is a single ITM as defined above. A set of (polynomially many) ITIs are called an *instance* of a protocol π if all ITIs in this set run the code of π and have the same SID. The ITIs in an instance of π which have a PID $\neq \perp$ are called the *(protocol) parties* of that instance of π.

Some of the ITIs in an instance of a protocol may be designated as *subroutines* of parties of that instance. As will be explained later, if an ITI P' is a subroutine of a party P then it can provide input to P and obtain subroutine output from P. A party P' that is a subroutine of a party P is called a *sub-party* of P and P is called a *calling party* of P'. A party of an instance of a protocol π that is not a subroutine of another party of that instance is called a *main party* of that instance of π.

In this work, we only consider cryptographic protocols that are *polynomially bounded*, i.e., which can be simulated by a single (non-interactive) Turing machine in probabilistic polynomial time.

Adversary and Environment. In the UC framework, an instance of a protocol π interacts with two additional ITIs called *adversary* and *environment.*[2] The adversary $\mathcal{A}$ has full control over the communication between protocol parties (i.e. messages to be written on the parties' incoming message tapes). More specifically, parties do not send each other messages directly but send the message (together with the address of the intended recipient) to $\mathcal{A}$ who is not obliged to forward this message to the intended recipient. If $\mathcal{A}$ does forward the message (unaltered) to the intended recipient, then one says that $\mathcal{A}$ *delivers* this message. The environment $\mathcal{Z}$ provides inputs to the main parties of the protocol and may receive outputs from the main parties. In addition, $\mathcal{Z}$ and $\mathcal{A}$ may interact freely with one another. It is stressed that $\mathcal{Z}$ has only access to the inputs and outputs of the main parties but neither has direct access to the communication between parties nor to the inputs and outputs of the subroutines. Furthermore, $\mathcal{A}$ has access only to the communication between parties but does not have access to their inputs and outputs.

Intuitively, the environment $\mathcal{Z}$ models protocols that provide input to and receive output from the instance of π. The adversary $\mathcal{A}$ models a malicious entity that attacks the protocol via the communication links, without having access to the protocol parties' (secret) local inputs and outputs.

For a more formal and detailed definition of the adversary and environment in the UC framework, see paragraph "UC Execution Experiment".

Ideal Functionalities and Ideal Protocols. An ideal functionality $\mathcal{F}$ is an ITI whose PID is set to be $\perp$. Typically, an ideal functionality $\mathcal{F}$ is a subroutine of multiple parties in an instance of a so-called *ideal protocol.* The ideal protocol with functionality $\mathcal{F}$ consists of N parties, called *dummy parties*, and the ideal functionality $\mathcal{F}$ which is a subroutine of all N parties. The interaction of an instance of the ideal protocol with functionality $\mathcal{F}$ with an adversary $\mathcal{A}$ and an environment $\mathcal{Z}$ is defined as follows:

- Each party forwards its input to $\mathcal{F}$.

[2]For the sake of concreteness, the adversary and environment are given the identities $(1, \perp)$ and $(0, \perp)$.

- $\mathcal{F}$ gives subroutine output to (a subset of) the parties. $\mathcal{F}$ is said to give *public delayed output* if $\mathcal{F}$ first sends the output to the adversary $\mathcal{A}$ along with the identity of the designated receiver. When $\mathcal{A}$ replies with the notification `deliver`, $\mathcal{F}$ sends the output to the designated party. $\mathcal{F}$ is said to give *private delayed output* if it behaves the same as for public delayed outputs, except that the adversary does not receive the output but only the identity of the designated receiver.
- Each party forwards the subroutine outputs coming from $\mathcal{F}$ as subroutine outputs to the environment $\mathcal{Z}$.
- The adversary $\mathcal{A}$ may interact with $\mathcal{F}$ (as specified by $\mathcal{F}$). (This communication models potential leakage of information on the protocol parties' inputs and outputs as well as a possibility of $\mathcal{A}$ to exert some influence, e.g. block an output or corrupt a dummy party, cf. "Party Corruption").
- All messages coming from the adversary are ignored by the parties.

Ideal protocols are used to formally define a cryptographic task. As an example, consider the task of computing the OR of N inputs (this is also known as "dining cryptographer's problem", cf. [Cha88]): the ideal functionality $\mathcal{F}_{\mathsf{OR}}$ takes all inputs, computes their OR, and then (successively) outputs the result to the parties. By design, correctness and input privacy hold trivially in the ideal protocol with functionality $\mathcal{F}_{\mathsf{OR}}$. Furthermore, since the parties provide their inputs to $\mathcal{F}_{\mathsf{OR}}$ without knowing the other parties' inputs, independence of inputs is also assured.

Non-Reactive and Reactive Functionalities. An ideal functionality is either *non-reactive* or *reactive*. A non-reactive functionality interacts with the parties in a single round, taking at most one input from each party and providing at most one output to each party. In contrast, a reactive functionality may receive inputs and provide outputs in multiple rounds, possibly maintaining state information between rounds.

Party Corruption. Unless specified otherwise (see below), party corruption is modeled by `corrupt` messages sent by the adversary $\mathcal{A}$ to the protocol parties. Upon receiving a `corrupt` message, a party outputs "corrupted" and sends its entire local state to $\mathcal{A}$. Also, all future inputs and subroutine outputs are forwarded to $\mathcal{A}$. Furthermore, $\mathcal{A}$ may instruct the corrupted party to send any message of his choice by writing the respective instruction on the corrupted parties' incoming message tape.

Party corruption is handled differently in ideal protocols. In these protocols, the dummy parties ignore all incoming messages from $\mathcal{A}$, in particular, `corrupt` messages. Party corruption is therefore handled solely by the ideal functionality $\mathcal{F}$. More specifically, $\mathcal{A}$ may corrupt a (dummy) party P by sending the message $(\texttt{corrupt}, P)$ to the ideal functionality $\mathcal{F}$. In this work, unless explicitly specified otherwise, we consider ideal functionalities that are *standard corruption*, which means that $\mathcal{F}$ proceeds as follows upon receiving a $(\texttt{corrupt}, P)$ message: $\mathcal{F}$ marks this party as `corrupted` and outputs "corrupted" to P. In the next activation, $\mathcal{F}$ sends to $\mathcal{A}$ all the inputs and outputs of P so far. In addition, from this point on, whenever $\mathcal{F}$ gets an input value v from P, it forwards v to

$\mathcal{A}$, who may then send a "modified input value" v' that overwrites v. Also, all output values intended for P are sent to $\mathcal{A}$ instead.

Static and Adaptive Corruption. In the UC framework, one distinguishes between *static* and *adaptive* corruptions. In the static corruption model, the adversary may only corrupt parties *prior* to the start of the protocol execution. In contrast, in the adaptive corruption model, the adversary may corrupt parties throughout the entire protocol execution.

UC Execution Experiment. An execution of a protocol σ with *adversary* $\mathcal{A}$ and an *environment* $\mathcal{Z}$ on input $a \in \{0,1\}^*$ and with security parameter $n \in \mathbb{N}$ is a run of a system of ITIs subject to the following restrictions:

- First, $\mathcal{Z}$ is activated on input $a \in \{0,1\}^*$.
- The first ITI to be invoked by $\mathcal{Z}$ is the adversary $\mathcal{A}$.
- $\mathcal{Z}$ may invoke a single instance of a *challenge protocol*, which is set to be σ by the experiment. The SID of σ is determined by $\mathcal{Z}$ upon invocation.
- $\mathcal{Z}$ may provide inputs to $\mathcal{A}$ and to the main parties of σ.
- The adversary $\mathcal{A}$ may send messages to the parties of σ as well as give subroutine outputs to $\mathcal{Z}$.
- Each party of σ may send messages to $\mathcal{A}$, provide inputs to its subroutines and give subroutine outputs to the parties of which it is a subroutine. Main parties may give subroutine outputs to $\mathcal{Z}$.
- The ITIs take turns during the execution experiment, i.e., whenever an ITI writes an allowed external write instruction, then the targeted ITI is activated and the sending ITI is suspended. If an ITI suspends its computation without writing an external write instruction or if it writes a disallowed external write instruction, then the environment $\mathcal{Z}$ is activated. (Note that this has the effect that at any point in time throughout the execution experiment only a single ITI is active.)
- At the end of the execution experiment, $\mathcal{Z}$ outputs a single bit.

Let $\mathrm{Exec}_{\mathsf{UC}}\big(\sigma, \mathcal{A}, \mathcal{Z}\big)(n, a)$ be the random variable defined as the output of the environment $\mathcal{Z}$ on input $a \in \{0,1\}^*$ and with security parameter $n \in \mathbb{N}$ after interacting with protocol σ and adversary $\mathcal{A}$ in the UC execution experiment.

Define $\mathrm{Exec}_{\mathsf{UC}}\big(\sigma, \mathcal{A}, \mathcal{Z}\big) = \big\{\mathrm{Exec}_{\mathsf{UC}}\big(\sigma, \mathcal{A}, \mathcal{Z}\big)(n, a)\big\}_{n \in \mathbb{N}, a \in \{0,1\}^*}$

In the UC framework, the security of a protocol is defined via the *real-ideal paradigm*. This means that a protocol π is deemed a secure realization of a given task if it is "as secure as" the ideal protocol with the functionality for that task. This is formalized by the notion of *UC emulation* which can be applied to any two protocols (not just real vs. ideal). Informally, a protocol π *UC-emulates* a protocol ϕ if for any adversary $\mathcal{A}$ interacting with π, there exists an adversary $\mathcal{S}$, called the *simulator*, interacting with ϕ such that no environment $\mathcal{Z}$ can distinguish between an interaction with π and $\mathcal{A}$ or ϕ and $\mathcal{S}$. A formal definition of this notion follows.

Definition 2.22 (UC Emulation). *Let π and ϕ be protocols. π is said to* UC-emulate *ϕ, denoted by $\pi \geq_{\mathsf{UC}} \phi$, if for every PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{S}$ such that for every PPT environment $\mathcal{Z}$ it holds that*

$$\mathrm{Exec}_{\mathsf{UC}}\big(\pi, \mathcal{A}, \mathcal{Z}\big) \overset{c}{\equiv} \mathrm{Exec}_{\mathsf{UC}}\big(\phi, \mathcal{S}, \mathcal{Z}\big)$$

Abusing notation, we will also write $\pi \geq_{\mathsf{UC}} \mathcal{F}$ if π UC-emulates the ideal protocol with functionality $\mathcal{F}$. In this case, we also say that π *UC-realizes* $\mathcal{F}$.

2.4.2 Standard Well-formed Ideal Functionalities

As argued in [Can+02], not all ideal functionalities can be UC-realized. [Can+02] therefore defined the class of "well-formed" functionalities and showed how to UC-realize every functionality in this restricted class. In the following, we recall their definition:

Well-Formed Functionalities. An ideal functionality is called *well-formed* if it consists of a "shell" and a "core". The core is an arbitrary PPT TM. The shell is a TM that acts as a "wrapper" in the following way: All incoming message are forwarded to the core except for `corrupt` messages. Furthermore, outputs generated by the core are forwarded by the shell. Moreover, an ideal functionality is *adaptively well-formed* if it consist of a shell and a core as described above and, in addition, the shell sends the random tape of the core to the adversary if all parties are corrupted at some activation.

Since [Can+02] worked in a different version of the UC framework, where the simulator is responsible for message delivery between the parties and the functionality and where party corruption is handled by the experiment instead of the functionality, we make the following additional restrictions:

Standard Functionalities. We call an ideal functionality $\mathcal{F}$ *standard* if $\mathcal{F}$ i) immediately notifies the adversary with the message $(\texttt{received}, P_i)$ upon receiving input from a party, and ii) is standard corruption[3], and iii) only gives (public or private) delayed outputs to the parties (except for "corrupted" outputs upon receiving `corrupt` messages from the adversary).

2.4.3 Some Important Functionalities

In the following, we list some important ideal functionalities that will be used in this work.

Definition 2.23 (Authenticated Message Transmission). *$\mathcal{F}_{\mathsf{auth}}$ proceeds as follows, running with parties S and R and an adversary $\mathcal{A}$:*

1. *Upon receiving an input $(\texttt{Send}, sid, S, R, m)$ from S, generate a public delayed output $(\texttt{Sent}, sid, S, R, m)$ to R.*

[3]Cf. Section 2.4.1 (paragraph "Party Corruption") for a definition of standard corruption.

2. *Upon receiving* $(\texttt{Corrupt}, sid, S, m')$ *from the adversary* $\mathcal{A}$*, and if* $(\texttt{Sent}, sid, S, R, m)$ *has not yet been delivered to* R*, then output* $(\texttt{Sent}, sid, S, R, m')$ *to* R*.*

Definition 2.24 (Common Reference String). $\mathcal{F}_{\mathsf{crs}}^{D}$ *proceeds as follows, running with parties* $P_1, \ldots, P_N$ *and an adversary* $\mathcal{A}$*, and parametrized by a distribution* D*:*

- *When activated for the first time on input* $(\texttt{value}, sid)$*, compute* $\texttt{crs} \leftarrow D(1^n)$*, and send a public delayed output* $(\texttt{crs}, sid)$ *to the activating party. In each other activation, send a public delayed output* $(\texttt{crs}, sid)$ *to the activating party.*

Definition 2.25 (Bit Commitments). $\mathcal{F}_{com}$ *proceeds as follows, running with parties* C *and* R *and an adversary* $\mathcal{A}$*:*

1. *Upon receiving an input* $(\texttt{Commit}, sid, C, R, b)$ *from* C*, where* $b \in \{0, 1\}$*, record the value* b *and generate a public delayed output* $(\texttt{Receipt}, sid, C, R)$ *to* R*. Ignore any subsequent* `Commit` *messages.*
2. *Upon receiving an input* $(\texttt{Unveil}, sid, C, R)$ *from* C*, proceed as follows: If some value* b *was previously recoded, then generated a public delayed output* $(\texttt{Unveil}, sid, C, R)$ *to* R*. Otherwise, do nothing.*

String Commitments: The ideal *string* commitment functionality $\mathcal{F}_{\mathsf{stcom}}$ is identical to $\mathcal{F}_{\mathrm{com}}$, except that the sender C can commit to a bitstring $s \in \{0, 1\}^n$.

Definition 2.26 (Oblivious Transfer). $\mathcal{F}_{\mathsf{OT}}$ *proceeds as follows, running with parties* S *and* R *and an adversary* $\mathcal{A}$*:*

1. *Upon receiving input* $(\texttt{Sender}, sid, S, R, b_0, b_1)$ *from* S*, where* $b_i \in \{0, 1\}$*, store* (b_0, b_1)*.*
2. *Upon receiving input* $(\texttt{Receiver}, sid, S, R, i)$ *from* R*, where* $i \in \{0, 1\}$*, wait until a tuple* (b_0, b_1) *is stored and then send a private delayed output* $(\texttt{Output}, sid, S, R, b_i)$ *to* R*.*

Definition 2.27 (Public Bulletin Board). $\mathcal{F}_{\mathsf{reg}}$ *proceeds as follows, running with parties* $P_1, \ldots, P_N$ *and an adversary* $\mathcal{A}$*:*

1. *Upon receiving a message* $(\texttt{register}, sid, v)$ *from party* P*,*
   send $(\texttt{registered}, sid, P, v)$ *to the adversary* $\mathcal{A}$*; upon receiving ok from* $\mathcal{A}$*, record the pair* (P, v)*. Otherwise, ignore the message.*
2. *Upon receiving a message* $(\texttt{retrieve}, sid, P_i)$ *from some party* P_j *(or the adversary* $\mathcal{A}$*), generate a public delayed output* $(\texttt{retrieve}, sid, P_i, v)$ *to* P_j*, where* $v = \perp$ *if no record* (P, v) *exists.*

Note that, in contrast to the usual definition in the literature (e.g. [CSV16]), $\mathcal{F}_{\mathsf{reg}}$ allows key revocation in this work.

2.4.4 Basic Properties of the Framework

In the following, we recall some important properties of the UC framework (cf. [Can01] for the proofs).

First, recall the definition of the dummy adversary:

Definition 2.28 (UC Emulation with Respect to the Dummy Adversary)**.** *The* dummy adversary $\mathcal{D}$ *is an adversary that when receiving a message* (sid, pid, m) *from the environment, sends* m *to the party with party identifier* pid *and session identifier* sid*, and that, when receiving* m *from the party with party identifier* pid *and session identifier* sid*, sends* (sid, pid, m) *to the environment.*

Let π *and* ϕ *be protocols.* π *is said to UC-emulate* ϕ with respect to the dummy adversary*, if there exists a PPT adversary* $\mathcal{S}_{\mathcal{D}}$ *such that for every PPT environment* $\mathcal{Z}$ *it holds that*

$$\mathrm{Exec}_{\mathsf{UC}}(\pi, \mathcal{D}, \mathcal{Z}) \overset{c}{\equiv} \mathrm{Exec}_{\mathsf{UC}}(\phi, \mathcal{S}_{\mathcal{D}}, \mathcal{Z})$$

A convenient property of the UC framework is that one need only consider the dummy adversary in security proofs because UC emulation is equivalent to UC emulation with respect to the dummy adversary.

Proposition 2.29 (Completeness of the Dummy Adversary)**.** *Let* π *and* ϕ *be protocols. Then,* π *UC-emulates* ϕ *if and only if* π *UC-emulates* ϕ *with respect to the dummy adversary.*

Another very useful property of the UC framework is that UC emulation is transitive. This property is important for modular analysis of protocols (cf. Section 2.4.5).

Proposition 2.30 (Transitivity)**.** *Let* π_1, π_2, π_3 *be protocols. If* $\pi_1 \geq_{\mathsf{UC}} \pi_2$ *and* $\pi_2 \geq_{\mathsf{UC}} \pi_3$*, then it holds that* $\pi_1 \geq_{\mathsf{UC}} \pi_3$*.*

2.4.5 Universal Composition

The central property of the UC framework is that its security notion is closed under protocol composition. More specifically, let ρ^{ϕ} be a protocol that makes subroutine calls to a polynomial number of instances of a protocol ϕ. The *universal composition theorem* guarantees that if a protocol π UC-emulates ϕ then one can replace all subroutine calls to instances of ϕ by subroutine calls to instances of π without losing security.

Theorem 2.31 (Universal Composition Theorem)**.** *Let* π, ϕ, ρ *be protocols. Then it holds that*[4]

$$\pi \underset{\mathsf{UC}}{\geq} \phi \implies \rho^{\pi} \underset{\mathsf{UC}}{\geq} \rho^{\phi}$$

The universal composition theorem has several important implications. First, it guarantees that the security properties of a UC-secure protocol remain valid even when multiple instances of that protocol are executed in an unknown environment. This property is called *concurrent security.*

[4]Actually, the universal composition theorem only holds if both π and ϕ are *subroutine respecting* protocols (cf. [Can01] for a definition). For simplicity, we ignore this subtlety in this work.

Another important consequence of the universal composition theorem is that it allows for *modular analysis* of protocols, i.e. the security of a composite protocol can be deduced from its components. This feature is essential for proving the security of complex protocols. To this end, the notion of *hybrid protocols* was introduced:

Hybrid Protocols. A protocol is said to be in the *$\mathcal{F}$-hybrid model* if it makes subroutine calls to the ideal protocol with functionality $\mathcal{F}$.

As an example, say one has a protocol ρ^{π} making subroutine calls to a commitment protocol π and wants to prove that ρ^{π} UC-emulates some ideal functionality $\mathcal{G}$. Instead of proving this statement "en bloc", one can separately prove that the (simpler) protocol $\rho^{\mathcal{F}_{\mathrm{com}}}$, which is in the $\mathcal{F}_{\mathrm{com}}$-hybrid model, UC-emulates $\mathcal{G}$ and that π UC-emulates $\mathcal{F}_{\mathrm{com}}$. The universal composition theorem and transitivity of UC emulation (cf. Proposition 2.30) then imply the desired statement that ρ^{π} UC-emulates $\mathcal{G}$.

Hybrid protocols are also used to model *trusted setup assumptions.* For instance, a protocol that requires a public bulletin board or a common reference string as a trusted setup is defined in the $\mathcal{F}_{\mathsf{reg}}$-hybrid model (cf. Definition 2.27) or $\mathcal{F}_{\mathsf{crs}}^{D}$-hybrid model (cf. Definition 2.24), respectively. A protocol is said to be in the *plain model* if it requires no trusted setup other than authenticated channels, i.e. if it is only in the $\mathcal{F}_{\mathsf{auth}}$-hybrid model.

Chapter 3

Non-malleability and CCA Security

This chapter is broken into two parts, both of which serve as a complement to the subsequent chapters. The first part deals with security notions related to *non-malleability* and *CCA security* for PKE schemes and contains no original work of the author. Several known relations between these notions are cited from the literature. PKE schemes satisfying such a notion will be used as a building block in the general MPC protocols presented in Chapter 5. The second part considers the corresponding security notions for commitment schemes, which will be important in the constructions presented in Chapter 4. As an additional contribution of this thesis, separations are proven between a variety of these security notions in this part.

The second part of this chapter is taken almost entirely from [Bro+18a], which was published at PKC 2018.

3.1 Notions for Public-Key Encryption Schemes

In this section, we briefly recall definitions and relations among several security notions related to non-malleablility and CCA security proposed for PKE schemes.

3.1.1 Variants of CCA Security

The notion of *indistinguishability under chosen ciphertext attacks* (IND-CCA security) was introduced in the work of [RS91]. Roughly speaking, a PKE scheme is IND-CCA-secure if an adversary is unable to break the security of the scheme even if he has access to an oracle that decrypts all ciphertexts except for the challenge ciphertext. IND-CCA secure PKE schemes have many applications. For instance, IND-CCA-secure PKE schemes were used as a building block for UC-realizing the ideal commitment functionality $\mathcal{F}_{\text{com}}$ (cf. Definition 2.25) in the *reusable* CRS-hybrid model [CF01]. Furthermore, IND-CCA secure PKE schemes are sufficient for realizing the public-key encryption functionality $\mathcal{F}_{\mathsf{PKE}}$ in the UC framework (in the case of static adversaries). In fact, UC-realizing $\mathcal{F}_{\mathsf{PKE}}$ is equivalent to IND-CCA security [Can01; CKN03]. In the following, we give a formal definition of this security notion.

Definition 3.1 (Indistinguishability under Adaptive Chosen Ciphertext Attacks)**.** *Let* $\mathsf{Exp}^{\mathsf{ind\text{-}cca}}_{\mathrm{PKE},\mathcal{A}(z)}(n)$ *denote the output of the following probabilistic experiment: At the beginning, the experiment generates a key pair* $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{Gen}_{\mathrm{PKE}}(1^n)$*. On input* 1^n*,* z *and* pk*, the adversary* $\mathcal{A}$ *chooses two messages* m_0, m_1 *and sends the tuple* (m_0, m_1) *to the experiment. The experiment then chooses a random bit* $b \leftarrow \{0,1\}$*, computes* $c^* \leftarrow \mathrm{Enc}(\mathrm{pk}, m_b)$ *and sends* c^* *to* $\mathcal{A}$*. At the end of the experiment,* $\mathcal{A}$ *sends a bit* $b' \in \{0,1\}$*. The experiment then outputs* 1 *if* $b = b'$*, and* 0 *otherwise. Througout the experiment* $\mathcal{A}$ *may send queries to the oracle* $\mathcal{O}_{\mathrm{Dec}(\mathrm{sk},\cdot)}$*.*

An adversary is called valid *if he only chooses messages* m_0, m_1 *such that* $|m_0| = |m_1|$ *and does not query* $\mathcal{O}_{\mathrm{Dec}(\mathrm{sk},\cdot)}$ *on* c^* *during the experiment.*

A public-key encryption scheme PKE *is* IND-CCA-secure *if for every valid PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $n \in \mathbb{N}, z \in \{0,1\}^*$ *it holds that*

$$\Pr[\mathsf{Exp}^{\mathsf{ind\text{-}cca}}_{\mathrm{PKE},\mathcal{A}(z)}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$$

Equivalent Formulation. Denote by $\mathrm{Output}^{\mathsf{ind\text{-}cca}}_{\mathrm{PKE},\mathcal{A}(z)}(n, b)$ the output of $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\mathsf{ind\text{-}cca}}_{\mathrm{PKE},\mathcal{A}(z)}(n)$, if the challenge bit is fixed to b.

A PKE scheme is IND-CCA-secure if and only if for every valid PPT adversary $\mathcal{A}$ there exists a negligible function negl such that for all $n \in \mathbb{N}, z \in \{0,1\}^*$ it holds that

$$\left|\mathrm{Output}^{\mathsf{ind\text{-}cca}}_{\mathrm{PKE},\mathcal{A}(z)}(n, 0) - \mathrm{Output}^{\mathsf{ind\text{-}cca}}_{\mathrm{PKE},\mathcal{A}(z)}(n, 1)\right| \leq \mathsf{negl}(n)$$

A variant of CCA security, put forward by [Bel+98], considers only adversaries that send a single parallel query to the decryption oracle. This notion is called *indistinguishability under parallel chosen ciphertext attacks* (IND-pCCA security). A formal definition follows.

Definition 3.2 (Indistinguishability under Parallel Chosen Ciphertext Attacks)**.** *Let* $\mathsf{Exp}^{\mathsf{ind\text{-}pcca}}_{\mathrm{PKE},\mathcal{A}(z)}(n)$ *denote the output of the following probabilistic experiment: At the beginning, the experiment generates a key pair* $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{Gen}_{\mathrm{PKE}}(1^n)$*. On input* 1^n*,* z *and* pk*, the adversary* $\mathcal{A}$ *chooses two messages* m_0, m_1 *and sends the tuple* (m_0, m_1) *to the experiment. The experiment then chooses a random bit* $b \leftarrow \{0,1\}$*, computes* $c^* \leftarrow \mathrm{Enc}(\mathrm{pk}, m_b)$ *and sends* c^* *to* $\mathcal{A}$*. At the end of the experiment,* $\mathcal{A}$ *sends a bit* $b' \in \{0,1\}$*. The experiment then outputs* 1 *if* $b = b'$*, and* 0 *otherwise. During the experiment* $\mathcal{A}$ *may send a single parallel query to the oracle* $\mathcal{O}_{\mathrm{Dec}(\mathrm{sk},\cdot)}$*.*

An adversary is called valid *if he only chooses messages* m_0, m_1 *such that* $|m_0| = |m_1|$ *and his parallel query to* $\mathcal{O}_{\mathrm{Dec}(\mathrm{sk},\cdot)}$ *does not contain* c^**.*

A public-key encryption scheme PKE *is* IND-pCCA-secure *if for every valid PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $n \in \mathbb{N}, z \in \{0,1\}^*$ *it holds that*

$$\Pr[\mathsf{Exp}^{\mathsf{ind\text{-}pcca}}_{\mathrm{PKE},\mathcal{A}(z)}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$$

Equivalent Formulation. Denote by $\mathrm{Output}^{\mathsf{ind\text{-}pcca}}_{\mathrm{PKE},\mathcal{A}(z)}(n,b)$ the output of $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\mathsf{ind\text{-}pcca}}_{\mathrm{PKE},\mathcal{A}(z)}(n)$, if the challenge bit is fixed to b.

A PKE scheme is IND-pCCA-secure if and only if for every valid PPT adversary $\mathcal{A}$ there exists a negligible function negl such that for all $n \in \mathbb{N}, z \in \{0,1\}^*$ it holds that

$$\left|\mathrm{Output}^{\mathsf{ind\text{-}pcca}}_{\mathrm{PKE},\mathcal{A}(z)}(n,0) - \mathrm{Output}^{\mathsf{ind\text{-}pcca}}_{\mathrm{PKE},\mathcal{A}(z)}(n,1)\right| \leq \mathsf{negl}(n)$$

3.1.2 Variants of Non-malleability

Informally, a PKE scheme is non-malleable if, given a ciphertext c under a public key pk encrypting a message m, an adversary is unable to produce another ciphertext $\tilde{c} \neq c$ (also under pk) such that $\tilde{c}$ decrypts to a message $\tilde{m}$ that is related to m. There are several variants of non-malleability for PKE schemes. In its basic form, called *non-malleability under chosen plaintext attacks* (NM-CPA security), the adversary is given no oracle to aid him [DDN91; Bel+98]. In contrast, the notion of *non-malleability under chosen ciphertext attacks* (NM-CCA security) grants the adversary access to a decryption oracle that may be queried on any ciphertext except for the challenge ciphertext [DDN00b]. In the following, we give formal definitions of these notions, paraphrasing from [Bel+98].

Definition 3.3 (Non-malleability under Chosen Plaintext Attacks)**.** *Denote by* $\mathsf{Exp}^{\mathsf{nm\text{-}cpa}}_{\mathrm{PKE},\mathcal{A}(z)}(n)$ *denote the output of the following probabilistic experiment: At the beginning, the experiment generates a key pair* $(\mathrm{pk},\mathrm{sk}) \leftarrow \mathrm{Gen}_{\mathrm{PKE}}(1^n)$*. On input* 1^n*,* z *and* pk*, the adversary* $\mathcal{A}$ *outputs a description of a message space, described by a sampling algorithm* $\mathcal{M}$*. The experiment then samples a message* $x \leftarrow \mathcal{M}$ *and computes* $y \leftarrow \mathrm{Enc}(\mathrm{pk},x)$ *and sends* y *to* $\mathcal{A}$*. The adversary then outputs a description of a relation* $\mathcal{R}$ *and a vector* $\mathbf{y}$*. The experiment then computes* $\mathbf{x} = \mathrm{Dec}(\mathrm{sk},\mathbf{y})$ *and outputs* 1 *if* $\mathcal{R}(x,\mathbf{x}) = 1$ *and* 0 *otherwise.*

Let $\widetilde{\mathsf{Exp}}^{\mathsf{nm\text{-}cpa}}_{\mathrm{PKE},\mathcal{A}(z)}(n)$ *denote the output of the following probabilistic experiment: At the beginning, the experiment generates a key pair* $(\mathrm{pk},\mathrm{sk}) \leftarrow \mathrm{Gen}_{\mathrm{PKE}}(1^n)$*. On input* 1^n*,* z *and* pk*, the adversary* $\mathcal{A}$ *outputs a description of a message space, described by a sampling algorithm* $\mathcal{M}$*. The experiment then samples a message* $x, \tilde{x} \leftarrow \mathcal{M}$ *and computes* $\tilde{y} \leftarrow \mathrm{Enc}(\mathrm{pk},\tilde{x})$ *and sends* $\tilde{y}$ *to* $\mathcal{A}$*. The adversary then outputs a description of a relation* $\mathcal{R}$ *and a vector* $\tilde{\mathbf{y}}$*. The experiment then computes* $\tilde{\mathbf{x}} = \mathrm{Dec}(\mathrm{sk},\tilde{\mathbf{y}})$ *and outputs* 1 *if* $\mathcal{R}(x,\tilde{\mathbf{x}}) = 1$ *and* 0 *otherwise.*

An adversary is called valid *if he only outputs a distribution* M *such that all strings having non-zero probability under* M *are of the same length, and no component of* $\tilde{\mathbf{y}}$ *equals* $\tilde{y}$*.*

A public-key encryption scheme PKE *is* NM-CPA-secure *if for every valid PPT adversary* $\mathcal{A}$ *there exists a negligible function* negl *such that for all* $n \in \mathbb{N}, z \in \{0,1\}^*$ *it holds that*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{nm\text{-}cpa}}_{\mathrm{PKE},\mathcal{A}(z)}(n) = 1] - \Pr[\widetilde{\mathsf{Exp}}^{\mathsf{nm\text{-}cpa}}_{\mathrm{PKE},\mathcal{A}(z)}(n) = 1]\right| \leq \mathsf{negl}(n)$$

Definition 3.4 (Non-malleability under Chosen Ciphertext Attacks)**.** *Denote by* $\mathsf{Exp}^{\mathsf{nm\text{-}cca}}_{\mathrm{PKE},\mathcal{A}(z)}(n)$ *denote the output of the following probabilistic experiment: At the beginning, the experiment generates a key pair* $(\mathrm{pk},\mathrm{sk}) \leftarrow \mathrm{Gen}_{\mathrm{PKE}}(1^n)$*.*

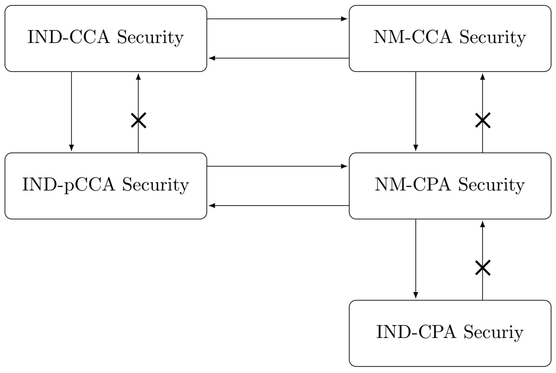


Figure 3.1: Relations among variants of non-malleability and CCA security (and IND-CPA security) proposed for PKE schemes.

On input 1^n, z *and* pk, *the adversary* $\mathcal{A}$ *outputs a description of a message space, described by a sampling algorithm* $\mathcal{M}$. *The experiment then samples a message* $x \leftarrow \mathcal{M}$ *and computes* $y \leftarrow \mathrm{Enc}(\mathrm{pk}, x)$ *and sends* y *to* $\mathcal{A}$. *The adversary then outputs a description of a relation* $\mathcal{R}$ *and a vector* $\mathbf{y}$. *The experiment then computes* $\mathbf{x} = \mathrm{Dec}(\mathrm{sk}, \mathbf{y})$ *and outputs* 1 *if* $\mathcal{R}(x, \mathbf{x}) = 1$ *and* 0 *otherwise. Througout the experiment* $\mathcal{A}$ *may send queries to the oracle* $\mathcal{O}_{\mathrm{Dec}(\mathrm{sk},\cdot)}$.

Let $\widetilde{\mathsf{Exp}}^{\mathsf{nm\text{-}cca}}_{\mathrm{PKE},\mathcal{A}(z)}(n)$ *denote the output of the following probabilistic experiment: At the beginning, the experiment generates a key pair* $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{Gen}_{\mathrm{PKE}}(1^n)$. *On input* 1^n, z *and* pk, *the adversary* $\mathcal{A}$ *outputs a description of a message space, described by a sampling algorithm* $\mathcal{M}$. *The experiment then samples a message* $x, \tilde{x} \leftarrow \mathcal{M}$ *and computes* $\tilde{y} \leftarrow \mathrm{Enc}(\mathrm{pk}, \tilde{x})$ *and sends* $\tilde{y}$ *to* $\mathcal{A}$. *The adversary then outputs a description of a relation* $\mathcal{R}$ *and a vector* $\tilde{\mathbf{y}}$. *The experiment then computes* $\tilde{\mathbf{x}} = \mathrm{Dec}(\mathrm{sk}, \tilde{\mathbf{y}})$ *and outputs* 1 *if* $\mathcal{R}(x, \tilde{\mathbf{x}}) = 1$ *and* 0 *otherwise. Througout the experiment* $\mathcal{A}$ *may send queries to the oracle* $\mathcal{O}_{\mathrm{Dec}(\mathrm{sk},\cdot)}$.

An adversary is called valid *if he only outputs a distribution* M *such that all strings having non-zero probability under* M *are of the same length, no component of* $\tilde{\mathbf{y}}$ *equals* $\tilde{y}$, *and he does not query* $\mathcal{O}_{\mathrm{Dec}(\mathrm{sk},\cdot)}$ *on* c^* *during the experiment.*

A public-key encryption scheme PKE *is* NM-CCA-secure *if for every valid PPT adversary* $\mathcal{A}$ *there exists a negligible function* negl *such that for all* $n \in \mathbb{N}, z \in \{0,1\}^*$ *it holds that*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{nm\text{-}cca}}_{\mathrm{PKE},\mathcal{A}(z)}(n) = 1] - \Pr[\widetilde{\mathsf{Exp}}^{\mathsf{nm\text{-}cca}}_{\mathrm{PKE},\mathcal{A}(z)}(n) = 1]\right| \leq \mathsf{negl}(n)$$

3.1.3 Relations

In this section, we very briefly recall the known relations between the security notions defined in the previous section (see Fig. 3.1 for an overview).

In [Bel+98] it is shown that IND-CCA security is equivalent to NM-CCA security. Furthermore, it also follows readily from some of the techniques in this work that NM-CPA security does not imply NM-CCA security and that IND-CPA security does not imply NM-CPA security.

Theorem 3.5 (IND-CCA $\Leftrightarrow$ NM-CCA)**.** *Every IND-CCA-secure PKE scheme is also NM-CCA-secure. Every NM-CCA-secure PKE scheme is also IND-CCA-secure.*

Theorem 3.6 (NM-CPA $\not\Rightarrow$ NM-CCA)**.** *If there exists an NM-CPA-secure PKE scheme, then there also exists an NM-CPA-secure PKE scheme that is not NM-CCA-secure.*

Theorem 3.7 (IND-CPA $\not\Rightarrow$ NM-CPA)**.** *If there exists an IND-CPA-secure PKE scheme, then there also exists an IND-CPA-secure PKE scheme that is not NM-CPA-secure.*

Theorem 3.5 shows the strength of IND-CCA security, establishing an equivalence between this notion and a strong form of non-malleability. Theorem 3.6 (together with Theorem 3.5) shows that mere non-malleability is indeed a weaker requirement than IND-CCA security. Note that while Theorem 3.7 implies that NM-CPA security is a strictly stronger notion than IND-CPA security, it can be shown that NM-CPA secure PKE schemes can be constructed given any IND-CPA secure PKE scheme without further assumptions [PSV06]. This can even be done in a black-box way [Cho+18].

In [BS99] it is shown that the notions of IND-pCCA security and NM-CPA security are equivalent.

Theorem 3.8 (IND-pCCA $\Leftrightarrow$ NM-CPA)**.** *Every IND-pCCA-secure PKE scheme is also NM-CPA-secure. Every NM-CPA-secure PKE scheme is also IND-pCCA-secure.*

Theorem 3.8 is very useful as it allows to work with Definition 3.2 instead of the more involved Definition 3.3 in security proofs. This theorem will later be used in Chapter 5.

3.2 Notions for Commitment Schemes

Informally, the (basic) setting of non-malleability for (tag-based) commitment schemes is the following: A malicious party $\mathcal{A}$, called man-in-the-middle, takes part in two sessions of a commitment protocol, playing the role of the sender in one session and that of the receiver in the other. $\mathcal{A}$'s goal is to produce a commitment to a value $\tilde{v}$ under a tag $\widetilde{tag}$ such that $\widetilde{tag}$ is different from the tag used in the session in which $\mathcal{A}$ is the receiver and $\tilde{v}$ is related to the value v to which $\mathcal{A}$ receives a commitment.

Several variants of non-malleability have been defined for commitment schemes in the literature, depending on the number and scheduling of the protocol sessions in which the man-in-the-middle $\mathcal{A}$ may participate. The most basic form is *stand-alone non-malleability* where $\mathcal{A}$ only participates in two sessions. The notion of *parallel non-malleability* considers adversaries that receive multiple commitments in parallel and commit to multiple values in parallel. In the setting of *concurrent non-malleability*, the adversary may receive and send multiple commitments in an arbitrary schedule determined by him.

Numerous works on non-malleable commitment schemes can be found in the literature (e.g. [CVZ10; Wee10; LP11; Goy+12; Cia+16; Goy11; Goy+14;

GPR16; Cia+17] to name a few). Non-malleable commitments have been used as a central building block for concurrently composable general MPC protocols both in the plain model (e.g. [LPV09; Gar+12; GKP17]) and based on additional setup assumptions (e.g. [LPV09]).

CCA security ("security against adaptive chosen commitment attacks") is a security notion that is closely related to non-malleability. CCA security generalizes the hiding property by granting the adversary access to an oracle that extracts committed values. A commitment scheme is said to be *CCA-secure* if it remains hiding even in the presence of an adversary who may send polynomially many queries in an arbitrary schedule to such an oracle. As with non-malleability, multiple variants of CCA security have been defined in the literature, depending on the number and scheduling of the queries the adversary may send to the oracle. *One-One CCA security* considers adversaries that may only send a single query consisting of exactly one commitment to the oracle. The notion of *parallel CCA security* considers adversaries that may query the oracle on a single query consisting of polynomially many commitments sent to the oracle in parallel.

CCA-secure commitment schemes were introduced by [CLP10] as a central building block for general MPC protocols in the plain model in the UC with super-polynomial helpers framework[1]. Since then, all general MPC protocols that have been constructed in the UC with super-polynomial helpers framework and Shielded Oracles framework[2] were based on commitment schemes satisfying a variant of CCA security (see [CLP10; CLP13; Kiy14; KMO14; LP12; Goy+15; HV16; Bro+17]). For instance, parallel CCA-secure commitment schemes and one-one CCA-secure commitment schemes were used as building blocks for several round-efficient general MPC protocols in the plain model in the UC with super-polynomial helpers framework [Kiy14; KMO14]. Furthermore, parallel CCA secure commitment schemes were also used as a building block for constant-round general MPC protocols in the plain model in the Shielded Oracles framework [Bro+17].[3]

3.2.1 Contribution

We settle the relations among a variety of security notions related to non-malleability and CCA security that have been proposed for commitment schemes in the literature (see Fig. 3.2). Our results show, in particular, that some of the known relations between notions defined for PKE schemes do not carry over to the case of commitment schemes. In particular, we show that parallel non-malleability and parallel CCA security are not equivalent, in contrast to the corresponding notions for PKE schemes (cf. Theorem 3.8).[4] In a little more detail, the results and techniques of this subchapter are the following:

Separation Results

We prove separations between multiple variants of non-malleability and CCA security proposed for commitment schemes (Theorems 3.17 to 3.28).

[1]Prior to [CLP10], the work of [PPV08] introduced the notion of *adaptively-secure commitments* which is a variant of CCA security for non-interactive perfectly binding commitments.
[2]Cf. Chapter 4 for a definition of these frameworks.
[3]The results of [Bro+17] are the subject of Chapter 4.
[4]It can be shown, however, that parallel non-malleability and parallel CCA security are equivalent for *non-interactive* commitment schemes by adapting the proof of Theorem 3.8 (cf. [Bro+18a].

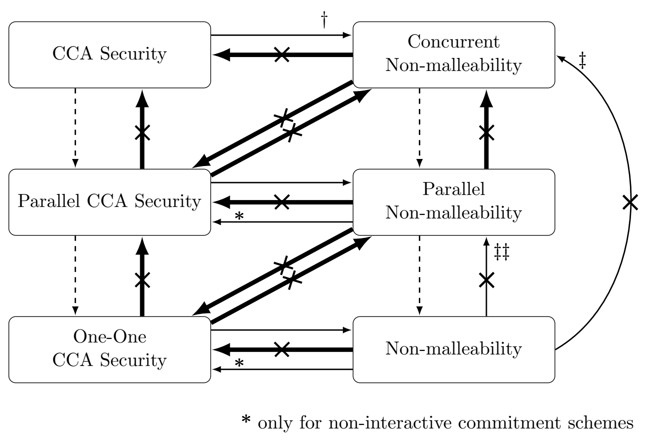

CCA Security
Concurrent Non-malleability
Parallel CCA Security
Parallel Non-malleability
One-One CCA Security
Non-malleability
†
‡
‡‡
*
* only for non-interactive commitment schemes

Figure 3.2: The relations between several security notions for commitment schemes. The dotted arrows indicate trivial implications. Markings indicate relations proven in the literature (see [PPV08] for † and [Cia+16] for ‡) or separating commitment schemes from the literature (see [LPV08] for ‡‡). The thick arrows indicate relations that are proven in this chapter.

Transformations We obtain all of our separation results using two generic transformations. Given two appropriate security notions X and Y from the class of security notions we compare in this chapter, these transformations take a commitment scheme that fulfills notion X and output a commitment scheme that still fulfills notion X but not notion Y. Both transformations are fully black-box and require no additional computational assumptions.

- *Puzzle-Solution-Approach (cf. Section 3.2.5)*: The first transformation is used for separations where Y is a CCA-related security notion. The key idea of this transformation is to expand a commitment scheme that fulfills a security notion X by a "puzzle phase" where the sender sends a specific computationally hard puzzle to the receiver. If the receiver answers with a correct solution, then the sender "gives up" and sends his input to the receiver who can then trivially win in the security game in this case. If the puzzle is tailored appropriately, then the expanded commitment scheme still fulfills notion X but fails to fulfill notion Y. Intuitively, this separation holds because an adversary in the Y-security game has access to an oracle that "breaks" the puzzle but an adversary in the X-security game does not.

- *Sharing-Approach (cf. Section 3.2.6)*: The second transformation is used for separations where Y is a variant of non-malleability. This transformation expands a given commitment scheme by adding a "share phase" in which the sender commits to two random shares of his input in a specific order. This is done in such a way that a man-in-the-middle adversary is able to forward these commitments to the receiver in his experiment. After the commit phase is over, these shares will be opened by the implicit oracle in the experiment and given to the distinguisher, who can then reconstruct the committed value.

Remark 3.9 (On Black-Box Separations). *We note that the separations proven in this chapter differ from* black-box separations. *Separating a security notion X from a security notion Y by a black-box separation means that one cannot construct a scheme satisfying X from a scheme satisfying Y in a* black-box manner. *Black-box separations are stronger than our separations. However, we note that one cannot achieve black-box separations between the security notions described in this chapter. This is because, given a (statistically binding) commitment scheme satisfying any of the security notions considered in this chapter, one can construct a commitment scheme satisfying any other security notion in this chapter in a black-box way. This can be shown as follows: First, each of the notions described in this chapter implies the standard hiding property for commitment schemes. Furthermore, given a commitment scheme that is binding and hiding, one can construct a one-way function in a black-box way [IL89]. Moreover, [Kiy14] showed how to construct a CCA-secure commitment scheme from any one-way function in a black-box way. Since CCA security implies any other notion described in this chapter, the statement follows. This transformation is, of course, highly inefficient and therefore only of theoretical interest. In particular, this transformation comes with a logarithmic blow-up in the round complexity, making it useless for constructing constant-round protocols (which is the goal in Chapter 4).*

3.2.2 Related Work

For the class of security notions for commitment schemes that are considered in this chapter, only a few relations are resolved. [PPV08] show that CCA security implies concurrent non-malleability. In [Cia+16] it is shown that the non-malleable commitment scheme from a preliminary version of [GPR16] is not concurrent non-malleable. [LPV08] construct a commitment scheme that separates non-malleability and parallel non-malleability. The remaining relations are, to the best of our knowledge, unsettled.

Notation. In the folllowing, we denote by $\mathsf{Com}_{tag}(v)$ a commitment to the value $v \in \{0,1\}^n$ under the tag $tag \in \{0,1\}^n$ using the (possibly interactive) commitment scheme Com. (Note that if we later use a phrase like "the sender sends $\mathsf{Com}_{tag}(v)$ to the receiver", we do not assume that the commitment scheme is non-interactive and hence consists of only one message. We rather use this formulation as an abbreviation for "the sender commits to v under the tag tag to the receiver using the commitment scheme Com".)

3.2.3 Variants of CCA Security

Roughly speaking, a tag-based commitment scheme Com is said to be CCA-secure, if the value committed to using a tag tag remains hidden even if the receiver has access to an oracle that "breaks" polynomially many commitments using a different tag $tag' \neq tag$. In this work, we only consider *committed value oracles* (oracles that return the committed value, cf. [LP12]) as opposed to *decommitment oracles* (oracles that return the entire decommitment information, cf. [CLP10]). In the following, we give a formal definition of CCA-secure commitment schemes.

Let Com be a tag-based, statistically binding commitment scheme. The CCA-oracle $\mathcal{O}_{\mathsf{cca}}$ for Com acts as follows in an interaction with an adversary $\mathcal{A}$: It participates with $\mathcal{A}$ in polynomially many sessions of the commit phase of Com as an honest receiver (the adversary determines the tag he wants to use at the start of each session). At the end of each session, if the session is valid, the oracle returns the unique value v committed to in the interaction; otherwise, it returns $\perp$. If a session has multiple valid committed values, the CCA-oracle also returns $\perp$. (The statistical binding property guarantees that this happens with only negligible probability.[5])

Let $\mathsf{Exp}^{\mathsf{cca}}_{\mathsf{Com},\mathcal{A}(z)}(n)$ denote the output of the following probabilistic experiment: Let $\mathcal{O}_{\mathsf{cca}}$ be the CCA-oracle for Com. The adversary has access to $\mathcal{O}_{\mathsf{cca}}$ during the entire course of the experiment. On input $1^n, z$, the adversary $\mathcal{A}^{\mathcal{O}_{\mathsf{cca}}}$ picks a tag tag and two strings v_0 and v_1 and sends (tag, v_0, v_1) to the experiment. The experiment then randomly selects a bit $b \leftarrow \{0,1\}$ and commits to v_b using the tag tag to $\mathcal{A}^{\mathcal{O}_{\mathsf{cca}}}$. Finally, $\mathcal{A}^{\mathcal{O}_{\mathsf{cca}}}$ sends a bit b' to the experiment, which outputs 1 if $b = b'$ and 0 otherwise.

An adversary is called *valid* if he only chooses strings v_0, v_1 such that $|v_0| = |v_1|$ and does not query $\mathcal{O}_{\mathsf{cca}}$ on a commitment that uses the challenge tag tag during the experiment.

Definition 3.10 (CCA-secure Commitment Schemes)**.** *Let* Com *be a tag-based, statistically binding commitment scheme. We say that* Com *is* CCA-secure*, if for every valid PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that for all $n \in \mathbb{N}, z \in \{0,1\}^*$ it holds that*

$$\mathsf{Pr}[\, \mathsf{Exp}^{\mathsf{cca}}_{\mathsf{Com},\mathcal{A}(z)}(n) = 1\,] \leq \frac{1}{2} + \mathsf{negl}(n)$$

Equivalent Formulation. Denote by $\mathrm{Output}^{\mathsf{cca}}_{\mathsf{Com},\mathcal{A}(z)}(n, b)$ the output of $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\mathsf{cca}}_{\mathsf{Com},\mathcal{A}(z)}(n)$ if the challenge bit is fixed to b.

A tag-based, statistically binding commitment scheme Com is CCA-secure if and only if for every valid PPT adversary $\mathcal{A}$ there exists a negligible function negl such that for all $n \in \mathbb{N}, z \in \{0,1\}^*$ it holds that

$$\left|\mathrm{Output}^{\mathsf{cca}}_{\mathsf{Com},\mathcal{A}(z)}(n, 0) - \mathrm{Output}^{\mathsf{cca}}_{\mathsf{Com},\mathcal{A}(z)}(n, 1)\right| \leq \mathsf{negl}(n)$$

The notion of *parallel CCA security* is a variant of CCA security where the adversary is restricted to a single parallel oracle query.

Formally, define the parallel CCA oracle $\mathcal{O}_{\mathsf{pcca}}$ for Com to be like $\mathcal{O}_{\mathsf{cca}}$, except that it only answers a single parallel query (consisting of sessions of the commit phase of Com. Note that the adversary determines the tags he wants to use at the start of the query). Define $\mathsf{Exp}^{\mathsf{pcca}}_{\mathsf{Com},\mathcal{A}(z)}(n)$ to be identical to $\mathsf{Exp}^{\mathsf{cca}}_{\mathsf{Com},\mathcal{A}(z)}(n)$, except that the adversary has access to $\mathcal{O}_{\mathsf{pcca}}$ instead of $\mathcal{O}_{\mathsf{cca}}$.

An adversary is called *valid* in this setting if he only chooses strings v_0, v_1 such that $|v_0| = |v_1|$ and his parallel query to $\mathcal{O}_{\mathsf{pcca}}$ does not contain a commitment that uses the challenge tag.

[5]We note that we could also assume the weaker property *strong computational binding* (cf. e.g. [Kiy14]). A commitment scheme satisfies the strong computational binding property if any (possibly malicious) PPT sender can generate a commitment that has more than one committed value with at most negligible probability.

Definition 3.11 (Parallel CCA-secure Commitment Schemes)**.** *Let* Com *be a tag-based, statistically binding commitment scheme. We say that* Com *is* parallel CCA-secure*, if for every valid PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $n \in \mathbb{N}, z \in \{0,1\}^*$ *it holds that*

$$\mathsf{Pr}[\,\mathsf{Exp}^{\mathsf{pcca}}_{\mathsf{Com},\mathcal{A}(z)}(n) = 1\,] \leq \frac{1}{2} + \mathsf{negl}(n)$$

Equivalent Formulation. Denote by $\mathrm{Output}^{\mathsf{pcca}}_{\mathsf{Com},\mathcal{A}(z)}(n,b)$ the output of $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\mathsf{pcca}}_{\mathsf{Com},\mathcal{A}(z)}(n)$ if the challenge bit is fixed to b.

A tag-based, statistically binding commitment scheme Com is parallel CCA-secure if and only if for every valid PPT adversary $\mathcal{A}$ there exists a negligible function negl such that for all $n \in \mathbb{N}, z \in \{0,1\}^*$ it holds that

$$\left|\mathrm{Output}^{\mathsf{pcca}}_{\mathsf{Com},\mathcal{A}(z)}(n,0) - \mathrm{Output}^{\mathsf{pcca}}_{\mathsf{Com},\mathcal{A}(z)}(n,1)\right| \leq \mathsf{negl}(n)$$

Another variant of CCA security is *one-one CCA security* where the adversary is restricted to a single query consisting of exactly one commitment.

Define the one-one CCA-oracle $\mathcal{O}_{\mathsf{1cca}}$ for Com to be like $\mathcal{O}_{\mathsf{cca}}$, except that it only answers a single query consisting of exactly one commitment. Define $\mathsf{Exp}^{\mathsf{1cca}}_{\mathsf{Com},\mathcal{A}(z)}(n)$ to be identical to $\mathsf{Exp}^{\mathsf{cca}}_{\mathsf{Com},\mathcal{A}(z)}(n)$, except that the adversary has access to $\mathcal{O}_{\mathsf{1cca}}$ instead of $\mathcal{O}_{\mathsf{cca}}$.

An adversary is called *valid* in this setting if he only chooses strings v_0, v_1 such that $|v_0| = |v_1|$ and his single query to $\mathcal{O}_{\mathsf{1cca}}$ does not use the challenge tag.

Definition 3.12 (One-One CCA-Secure Commitment Schemes)**.** *Let* Com *be a tag-based, statistically binding commitment scheme. We say that* Com *is* one-one CCA-secure*, if for every valid PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* $n \in \mathbb{N}, z \in \{0,1\}^*$ *it holds that*

$$\mathsf{Pr}[\,\mathsf{Exp}^{\mathsf{1cca}}_{\mathsf{Com},\mathcal{A}(z)}(n) = 1\,] \leq \frac{1}{2} + \mathsf{negl}(n)$$

3.2.4 Variants of Non-malleability

We next give a definition of (stand-alone) non-malleable commitment schemes that is essentially a game-based variant of the definition of [GPR16]. It is easy to see that the notion in [GPR16] and our notion are equivalent. Using a game-based variant of [GPR16] makes it easier to compare this notion with CCA security (which is game-based).

Let $\mathsf{Exp}^{\mathsf{nm}}_{\mathsf{Com},\mathcal{A}(z),\mathcal{D}}(n)$ denote the output of the following probabilistic experiment: On input $1^n, z$, the man-in-the-middle adversary $\mathcal{A}$ picks a tag tag and two strings v_0 and v_1 and sends (tag, v_0, v_1) to the sender S. The sender S then chooses a random bit $b \leftarrow \{0,1\}$. $\mathcal{A}$ then interacts with S and a receiver R, receiving a commitment $\mathsf{Com}_{tag}(v_b)$ from S and attempting to commit to a value $\tilde{v}_b$ using a tag $\widetilde{tag}$ of his choice in the interaction with R. $\mathcal{A}$ controls the scheduling of the messages. At the end of this interaction, $\mathcal{A}$ outputs his view $view_{\mathcal{A}}$ and R outputs the value $\tilde{v}_b$. Note that R has implicit access to a super-polynomial-time oracle $\mathcal{O}$ that extracts the committed value of the received commitment for him and that the adversary's view contains the non-uniform input z, the random coins used by the adversary and a transcript of all messages received by the adversary. After the interaction has finished, the distinguisher $\mathcal{D}$ gets the view $view_{\mathcal{A}}$ of the adversary and the value $\tilde{v}_b$ as input and outputs a bit b'. The experiment outputs 1 if $b = b'$ and 0 otherwise.

An adversary is called *valid* if he only chooses strings v_0, v_1 such that $|v_0| = |v_1|$ and does not use the tag $\widetilde{tag} = tag$ in his interaction with the receiver R.

Definition 3.13 (Non-malleable Commitment Schemes). *Let* Com *be a tag-based, statistically binding commitment scheme.* Com *is* non-malleable *if for every valid PPT man-in-the-middle adversary $\mathcal{A}$, for every PPT distinguisher $\mathcal{D}$, there exists a negligible function* negl *such that for all $n \in \mathbb{N}, z \in \{0,1\}^*$ it holds that*

$$\Pr[\,\mathsf{Exp}^{\mathsf{nm}}_{\mathsf{Com},\mathcal{A}(z),\mathcal{D}}(n) = 1\,] \leq \frac{1}{2} + \mathsf{negl}(n)$$

[LPV08] proposed a more general variant of non-malleability called *concurrent non-malleability* (CNM) where the man-in-the-middle adversary participates in m left and m right interactions, where $m = poly(n)$.

More specifically, the experiment for concurrent non-malleability is the same as for (stand-alone) non-malleability, except that now the adversary $\mathcal{A}$ sends a tuple $(\vec{v}^0, \vec{v}^1)$, where $\vec{v}^0 = (v^0_1, \ldots, v^0_m)$ and $\vec{v}^1 = (v^1_1, \ldots, v^1_m)$, to the sender S. The sender S then chooses a random bit $b \leftarrow \{0,1\}$. $\mathcal{A}$ then interacts with S and a receiver R, receiving commitments to values $v^b_1, \ldots, v^b_m$ with tags $tag_1, \ldots, tag_m$ of his choice from the sender S and attempting to commit to values $\tilde{v}^b_1, \ldots, \tilde{v}^b_m$ using tags $\widetilde{tag}_1, \ldots, \widetilde{tag}_m$ of his choice in the interactions with R ($\mathcal{A}$ determines the tag he wants to use at the start of each session). $\mathcal{A}$ has control over the scheduling of the messages. Let $\mathsf{Exp}^{\mathsf{cnm}}_{\mathsf{Com},\mathcal{A}(z),\mathcal{D}}(n)$ denote the output of this experiment.

An adversary is called *valid* in this setting if he only chooses vectors $\vec{v}^0 = (v^0_1, \ldots, v^0_m)$ and $\vec{v}^1 = (v^1_1, \ldots, v^1_m)$ such that $|v^0_i| = |v^1_i|$ for all $i \in \{1, \ldots, m\}$ and does not use any of the tags $tag_1, \ldots, tag_m$ that are used in the left interactions in his interactions with the receiver R.

Definition 3.14 (Concurrent Non-Malleable Commitment Schemes). *Let* Com *be a tag-based, statistically binding commitment scheme.* Com *is* concurrent non-malleable *if for every valid PPT man-in-the-middle adversary $\mathcal{A}$, for every PPT distinguisher $\mathcal{D}$, there exists a negligible function* negl *such that for all $n \in \mathbb{N}, z \in \{0,1\}^*$ it holds that*

$$\Pr[\,\mathsf{Exp}^{\mathsf{cnm}}_{\mathsf{Com},\mathcal{A}(z),\mathcal{D}}(n) = 1\,] \leq \frac{1}{2} + \mathsf{negl}(n)$$

Finally, *parallel non-malleability* (PNM) is another variant of non-malleabilty where the adversary $\mathcal{A}$ receives m commitments in *parallel* from the sender S and participates in m parallel interactions with the receiver R.[6] Let $\mathsf{Exp}^{\mathsf{pnm}}_{\mathsf{Com},\mathcal{A}(z),\mathcal{D}}(n)$ denote the output of this experiment.

An adversary is called *valid* in this setting if he only chooses vectors $\vec{v}^0 = (v^0_1, \ldots, v^0_m)$ and $\vec{v}^1 = (v^1_1, \ldots, v^1_m)$ such that $|v^0_i| = |v^1_i|$ for all $i \in \{1, \ldots, m\}$ and does not use any of the tags that are used in the m parallel left interactions in his m parallel interactions with the receiver R.

[6] [Gar+16] considered a notion where the adversary receives multiple commitments in parallel from S but is allowed to send commitments in an *arbitrary* scheduling in the interactions with R. We note that our separation results also hold for this variant.

Definition 3.15 (Parallel Non-Malleable Commitment Schemes)**.** *Let* Com *be a tag-based, statistically binding commitment scheme.* Com *is* parallel non-malleable *if for every valid PPT man-in-the-middle adversary* $\mathcal{A}$*, for every PPT distinguisher* $\mathcal{D}$*, there exists a negligible function* negl *such that for all* $n \in \mathbb{N}, z \in \{0,1\}^*$ *it holds that*

$$\mathsf{Pr}[\,\mathsf{Exp}^{\mathsf{pnm}}_{\mathsf{Com},\mathcal{A}(z),\mathcal{D}}(n) = 1\,] \leq \frac{1}{2} + \mathsf{negl}(n)$$

3.2.5 First Transformation (Puzzle-Solution Approach)

In this section, we describe the first transformation in this chapter. We call this approach the *puzzle-solution approach* because the general idea is to expand a commitment scheme by a "puzzle phase" that is executed at the beginning. Let X and Y be security notions for commitment schemes for which we want to show that X does not imply Y. For the first transformation, Y will always be a CCA-related security notion. Let $\mathcal{O}_X$ be the oracle an adversary can use in the security game for the notion X. Let $\mathcal{O}_Y$ be the oracle an adversary can use in the security game for the notion Y (note that these oracles may be the "empty oracle" which returns $\perp$ for each query).

Construction

Let Com be a (possibly interactive) commitment scheme. We will sometimes call Com the base commitment scheme.

Using Com, one can define the separating commitment scheme, which we will denote by Com'. We define Com' as output of a transformation $\mathsf{PComGen}$ that gets a base commitment scheme, a number $l \in \mathbb{N}$ and a string $\mathsf{sch} \in \{seq, par\}$ as input, i.e., $\mathsf{Com}' \leftarrow \mathsf{PComGen}(\mathsf{Com}, l, \mathsf{sch})$.

In the commitment scheme Com' the sender S, who wants to commit to a value v given a tag tag, first sends a "puzzle" to the receiver R and, depending on whether R solves the puzzle or not, either sends v as plaintext or commits to v using the base commitment scheme Com. The puzzle consists of l commitments to random messages (using Com) that are either sent in parallel (if $\mathsf{sch} = par$) or sequentially (if $\mathsf{sch} = seq$) to R. More specifically, the sender randomly generates l tags of length k and l values also of length k, i.e., $(tag_p^1, \dots, tag_p^l) \xleftarrow{\$} \left(\{0,1\}^k\right)^l$, $(w_1, \dots, w_l) \xleftarrow{\$} \left(\{0,1\}^k\right)^l$.

If $\mathsf{sch} = par$, the sender commits in *parallel* to $(w_1, \dots, w_l)$ under the tags $(tag_p^1, \dots, tag_p^l)$ to the receiver. The receiver then answers by simply guessing, i.e., sends a tuple of random values $(w_1', \dots, w_l')$. The sender then checks if the receiver's guess is correct, i.e. if for all $i \in \{1, \dots, l\}$ it holds that $w_i = w_i'$. If this is the case, S sends v as plaintext to the receiver. If it does not hold, S commits to v using the tag tag and the commitment scheme Com to R.

If $\mathsf{sch} = seq$, the sender *sequentially* commits to $(w_1, \dots, w_l)$ using the tags $(tag_p^1, \dots, tag_p^l)$ to the receiver. More specifically, he first commits to w_1 using the tag tag_p^1 and the commitment scheme Com and waits for a possible solution. The receiver R then sends a random value w_1' to S. If the solution is incorrect, then S commits to v using the tag tag and the base commitment scheme Com to R. Otherwise, he continues the puzzle phase by sending the second puzzle commitment, i.e., $\mathsf{Com}_{tag_p^2}(w_2)$, to R and again waits for a the possible solution.

The receiver R then sends another random value w_2' to S. If the solution is incorrect, then S commits to v using the tag *tag* and the commitment scheme Com. Otherwise, he continues by sending the third puzzle commitment and so forth. If R has correctly solved all l puzzle commitments, S sends v as plaintext to the receiver.

Proof Strategy

To prove that X does not imply Y, one shows that the commitment scheme Com' still fulfills X if the base commitment scheme Com fulfills X, but not Y. This is done by selecting l and sch in such a way that the puzzle can be solved with $\mathcal{O}_Y$ but not with $\mathcal{O}_X$.

First, consider the following definition:

Definition 3.16 ($\mathcal{O}$-one-way Commitment Schemes)**.** *Let* Com *be a tag-based commitment scheme and* $\mathcal{O}$ *a specific oracle for it.*

Let $\mathsf{Exp}^{\mathsf{ow}}_{\mathsf{Com},\mathcal{A}(z),\mathcal{O}}(n)$ *denote the output of the following probabilistic experiment: The experiment generates a random value v and a random tag* tag*, i.e., $v \leftarrow \{0,1\}^n$, $tag \leftarrow \{0,1\}^n$. It then sends the commitment* $\mathsf{Com}_{tag}(v)$ *as challenge to the adversary $\mathcal{A}$. On input 1^n, z, the adversary sends a value v' to the experiment which outputs* 1 *if $v = v'$ and* 0 *otherwise. During the experiment the adversary has access to the oracle $\mathcal{O}$.*

An adversary $\mathcal{A}$ is called valid *if he does not query the oracle $\mathcal{O}$ on a commitment that uses the challenge tag* tag *during the experiment.*

We say that Com *is* $\mathcal{O}$-one-way*, if for every valid PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that for all $n \in \mathbb{N}, z \in \{0,1\}^*$ it holds that*

$$\Pr[\,\mathsf{Exp}^{\mathsf{ow}}_{\mathsf{Com},\mathcal{A}(z),\mathcal{O}}(n) = 1\,] \leq \mathsf{negl}(n)$$

The above definition can be instantiated with various oracles. For example, $\mathcal{O}_{\mathsf{cca}}$-one-wayness describes a security notion where the one-way adversary has access to the CCA-oracle for the given commitment scheme. It is easy to see that CCA security implies $\mathcal{O}_{\mathsf{cca}}$-one-wayness. Similarly, parallel CCA security implies $\mathcal{O}_{\mathsf{pcca}}$-one-wayness and one-one CCA security implies $\mathcal{O}_{\mathsf{1cca}}$-one-wayness. Also note that all variants of non-malleability imply ε-one-wayness for the empty oracle ε, which just returns $\perp$ for each query.

The proof strategy now proceeds as follows:

Show that Com' is not Y-secure. One can construct an adversary $\mathcal{A}$ that breaks the Y-security of Com'. $\mathcal{A}$ simply forwards the puzzle to the oracle $\mathcal{O}_Y$, which returns the correct solution with overwhelming probability. Once $\mathcal{A}$ has the correct solution, he can trivially win in the security game for Y.

The probability that $\mathcal{A}$ wins the game is overwhelming because the only possibilities how $\mathcal{A}$ can lose are: 1) the oracle solves the puzzle it gets before the query, 2) a session with the oracle has multiple valid committed values and $\mathcal{O}_Y$ thus returns $\perp$, 3) during the execution the adversary queries the oracle on a commitment that uses the challenge tag (which happens if a puzzle commitment uses the challenge tag). Since one can show that each possibility occurs only with negligible probability, the overall winning probability of $\mathcal{A}$ is overwhelming.

Show that Com' is X-secure (under the assumption that Com is X-secure). Let $\mathcal{A}$ be an adversary against Com' in the security game for X, who wins the game with non-negligible advantage. Depending whether or not $\mathcal{A}$ solves at least one puzzle[7] in the security game for X, one has to distinguish two cases. For each case one builds an adversary who breaks the X-security of the commitment scheme Com.

Case 1: $\mathcal{A}$ solves at least one puzzle. In this case, one constructs an adversary $\mathcal{B}_1$ against the $\mathcal{O}_X$-one-wayness of Com. Recall that X-security implies $\mathcal{O}_X$-one-wayness for our cases. We denote by k the number of challenge commitments $\mathcal{A}$ expects. Since each of the k corresponding puzzles contains l commitments, $\mathcal{A}$ expects $m = l \cdot k$ puzzle commitments in total. The strategy of $\mathcal{B}_1$ is then first to randomly generate $m-1$ puzzle values and tags and to randomly select a $j \in \{1, \dots, m\}$. After $\mathcal{B}_1$ has received the challenge $\mathsf{Com}_{tag}(v)$ from the experiment, he starts to send $\mathcal{A}$ the puzzle(s). For all puzzle commitments except the j^{th} one he uses the honestly generated values and tags. As j^{th} puzzle commitment he uses the challenge commitment $\mathsf{Com}_{tag}(v)$. After $\mathcal{A}$ has sent the solution to the j^{th} puzzle commitment (i.e. $\mathcal{B}_1$'s challenge commitment), $\mathcal{B}_1$ terminates the simulation of $\mathcal{A}$ and sends $\mathcal{A}$'s solution to the j^{th} puzzle commitment as his own solution to the experiment.

If $\mathcal{A}$ asks his oracle $\mathcal{O}_X$ during the game, $\mathcal{B}_1$ sends random answers in the puzzle phase (to simulate the oracle) and forwards the actual oracle query to his own oracle $\mathcal{O}_X$. Note that it is possible that $\mathcal{B}_1$'s experiment returns $\perp$ at the end of the experiment. This happens if one of $\mathcal{A}$'s oracle queries contains a tag that equals $\mathcal{B}_1$'s challenge tag. This case may occur with non-negligible probability because the challenge tags of $\mathcal{A}$ and $\mathcal{B}_1$ are not necessarily identical. Fortunately, the opposite event also occurs with non-negligible probability.

The adversary $\mathcal{B}_1$ thus wins his game if $\mathcal{A}$ solves the puzzle commitment that is the challenge and $\mathcal{A}$'s oracle queries do not contain the challenge tag.

Case 2: $\mathcal{A}$ solves none of the puzzles. In this case one constructs an adversary $\mathcal{B}_2$ against the X-security of Com. The strategy of $\mathcal{B}_2$ is to send random puzzle(s) to $\mathcal{A}$, who fails to solve them (by assumption). After the puzzle phase, $\mathcal{B}_2$ forwards his own challenge to $\mathcal{A}$. The adversary $\mathcal{B}_2$ also forwards $\mathcal{A}$'s solution as his own solution to the experiment.

If $\mathcal{A}$ asks his oracle $\mathcal{O}_X$ during the game, $\mathcal{B}_2$ sends random answers in the puzzle phase (to simulate the oracle) and forwards the actual oracle query to his own oracle $\mathcal{O}_X$. Here, the challenge tags of $\mathcal{A}$ and $\mathcal{B}_2$ are always identical (because $\mathcal{B}_2$ forwards it to his experiment), so the possibility of $\mathcal{B}_2$'s experiment outputting $\perp$ is not a problem in this case.

The adversary $\mathcal{B}_2$ thus wins his game if $\mathcal{A}$ wins his own game and solves no puzzle.

[7]Note that for example in the concurrent non-malleability security game multiple puzzles (with $l = 1$ for each puzzle) are sent (one for each session).

A Concrete Example: CNM Does Not Imply CCA Security

In this section, we apply the puzzle-solution approach to separate the notion of CCA security from the notion of concurrent non-malleability.[8] To this end, we define Com' as $\mathsf{Com}' \leftarrow \mathsf{PComGen}(\mathsf{Com}, 1, seq)$ where Com is a statistically binding, concurrent non-malleable commitment scheme. The puzzle hence consists of just one commitment (thus the scheduling does not matter in this case). We follow the proof strategy described in Sec. 3.2.5.

Theorem 3.17 (CNM $\not\Rightarrow$ CCA). *If* Com *is a statistically binding, concurrent non-malleable commitment scheme, then* $\mathsf{Com}' \leftarrow \mathsf{PComGen}(\mathsf{Com}, 1, seq)$ *is also statistically binding and concurrent non-malleable but not CCA-secure.*

Proof. The statistical binding property of Com' follows readily from the statistical binding property of the underlying commitment scheme Com. In the following, we prove that Com' is concurrent non-malleable but not CCA-secure.[9]

Claim 1: Com' is not CCA-secure. We show that we can build a CCA-adversary $\mathcal{A}$, such that $\mathcal{A}$ wins the CCA security game for the commitment scheme Com' with non-negligible advantage.

The CCA-adversary $\mathcal{A}$ acts as depicted in Fig. 3.3. His strategy is to let the oracle solve the puzzle he got from the experiment and to hence get the challenge as plaintext. There are three possibilities how $\mathcal{A}$ can lose the game:

- The oracle solves the puzzle, i.e., $y = w_p^*$.
- The puzzle tag equals the challenge tag, i.e., $tag = tag_p$ (in that case the experiment returns $\perp$ as result instead of a bit).
- The query sent to the oracle has more than one valid opening (in that case the oracle returns $w'_p = \perp$).

The first possibility occurs with probability $1/2^n$ because the oracle uniformly selects a solution. The second possibility also occurs with probability $1/2^n$ because the puzzle tag is uniformly selected. The third possibility occurs with negligible probability, which we denote by negl, because Com is by assumption statistically binding. Thus, $\mathcal{A}$'s success probability is overwhelming:

$$\begin{aligned}\Pr[\mathsf{Exp}^{\mathsf{cca}}_{\mathsf{Com}',\mathcal{A}}(n) = 1] &\geq 1 - \frac{1}{2^n} - \frac{1}{2^n} - \mathsf{negl}(n)\\ &= 1 - \frac{1}{2^{n-1}} - \mathsf{negl}(n)\end{aligned}$$

Claim 2: Com' is concurrent non-malleable. Let us assume Com' is not concurrent non-malleable. Then we show that Com is also not concurrent non-malleable. Consider an adversary $\mathcal{A}$ and distinguisher $\mathcal{D}_\mathcal{A}$ such that $\mathcal{D}_\mathcal{A}$ wins in the concurrent non-malleability security game for the commitment scheme

[8]While the separation of CCA security from concurrent non-malleability is not very surprising, we have nonetheless chosen to give a full proof for this separation. This is because this proof is one of the easier applications of our puzzle-solution approach and therefore (hopefully) a good example for the reader.

[9]For ease of notation, we omit the (non-uniform) input z of the adversary and distinguisher. The proof can be easily adapted to include this input.

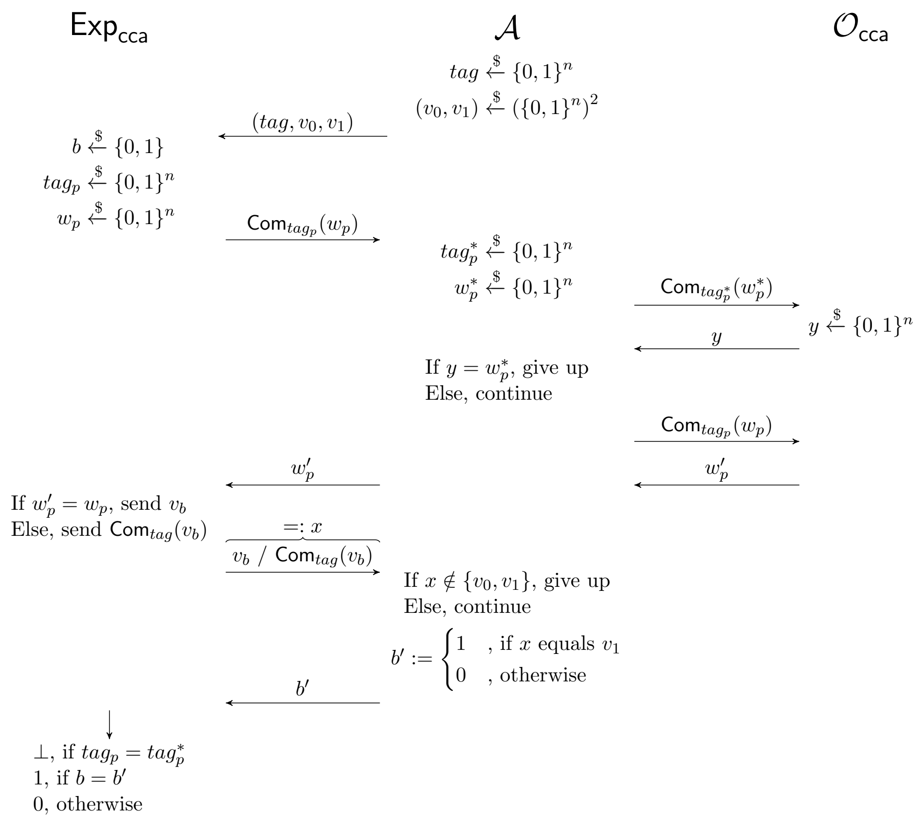

$\mathsf{Exp}_{\mathsf{cca}}$
$\mathcal{A}$
$\mathcal{O}_{\mathsf{cca}}$
$tag \xleftarrow{\$} \{0,1\}^n$
$(v_0, v_1) \xleftarrow{\$} (\{0,1\}^n)^2$
(tag, v_0, v_1)
$b \xleftarrow{\$} \{0,1\}$
$tag_p \xleftarrow{\$} \{0,1\}^n$
$w_p \xleftarrow{\$} \{0,1\}^n$
$\mathsf{Com}_{tag_p}(w_p)$
$tag_p^* \xleftarrow{\$} \{0,1\}^n$
$w_p^* \xleftarrow{\$} \{0,1\}^n$
$\mathsf{Com}_{tag_p^*}(w_p^*)$
y
$y \xleftarrow{\$} \{0,1\}^n$
If $y = w_p^*$, give up
Else, continue
$\mathsf{Com}_{tag_p}(w_p)$
w'_p
w'_p
If $w'_p = w_p$, send v_b
Else, send $\mathsf{Com}_{tag}(v_b)$
$\overbrace{v_b \ / \ \mathsf{Com}_{tag}(v_b)}^{=:\, x}$
If $x \notin \{v_0, v_1\}$, give up
Else, continue
$b' := \begin{cases} 1 & \text{, if } x \text{ equals } v_1 \\ 0 & \text{, otherwise} \end{cases}$
b'
$\perp$, if $tag_p = tag_p^*$
1, if $b = b'$
0, otherwise

Figure 3.3: Graphical depiction of the behavior of the adversary $\mathcal{A}$ in the CCA security game for the commitment scheme Com'. Note that $w'_p \in \{w_p, \perp\}$ is either the unique committed value w_p or, if the commitment has more than one valid opening, $\perp$.

Com' with advantage $\mathsf{Adv}^{\mathsf{cnm}}_{\mathsf{Com}',\mathcal{A},\mathcal{D}_\mathcal{A}}(n)$. Let $m = poly(n)$, where n is the security parameter, be the number of concurrent commitment sessions initiated by the sender in the concurrent non-malleability security game for Com'. Then we can split up $\mathcal{D}_\mathcal{A}$'s success probability as follows:

$$\begin{aligned} \Pr[\mathsf{Exp}^{\mathsf{cnm}}_{\mathsf{Com}',\mathcal{A},\mathcal{D}_\mathcal{A}}(n) = 1] = &\Pr[\mathsf{Exp}^{\mathsf{cnm}}_{\mathsf{Com}',\mathcal{A},\mathcal{D}_\mathcal{A}}(n) = 1 \wedge \exists i : \mathcal{A} \text{ solves puzzle } i] \\ &+ \Pr[\mathsf{Exp}^{\mathsf{cnm}}_{\mathsf{Com}',\mathcal{A},\mathcal{D}_\mathcal{A}}(n) = 1 \wedge \nexists i : \mathcal{A} \text{ solves puzzle } i] \end{aligned} \tag{3.1}$$

Hence, in the following it suffices to consider that $\mathcal{A}$ wins and

- Case 1: $\mathcal{A}$ solves at least one of the m puzzles.
- Case 2: $\mathcal{A}$ solves none of the m puzzles.

Case 1: $\mathcal{A}$ solves at least one of the m puzzles. Using $\mathcal{A}$ we construct an adversary $\mathcal{B}_1$ against the ε-one-wayness (for the empty oracle ε) of the commitment scheme Com. The adversary $\mathcal{B}_1$ acts as depicted in Fig. 3.4 in the ε-one-way security

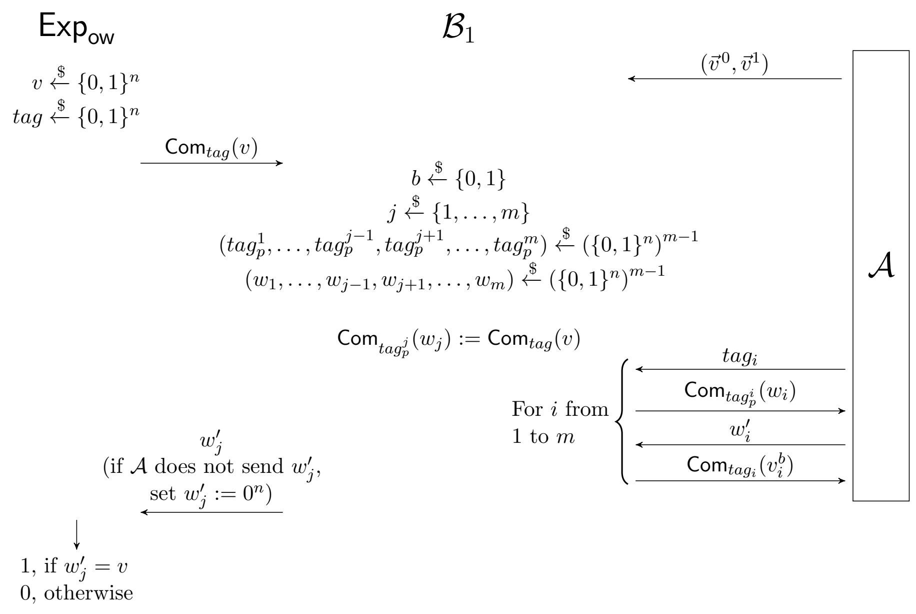

$\mathsf{Exp}_{\mathsf{ow}}$
$\mathcal{B}_1$
$(\vec{v}^0, \vec{v}^1)$
$v \xleftarrow{\$} \{0,1\}^n$
$tag \xleftarrow{\$} \{0,1\}^n$
$\mathsf{Com}_{tag}(v)$
$b \xleftarrow{\$} \{0,1\}$
$j \xleftarrow{\$} \{1,\dots,m\}$
$(tag_p^1,\dots,tag_p^{j-1},tag_p^{j+1},\dots,tag_p^m) \xleftarrow{\$} (\{0,1\}^n)^{m-1}$
$(w_1,\dots,w_{j-1},w_{j+1},\dots,w_m) \xleftarrow{\$} (\{0,1\}^n)^{m-1}$
$\mathcal{A}$
$\mathsf{Com}_{tag_p^j}(w_j) := \mathsf{Com}_{tag}(v)$
tag_i
$\mathsf{Com}_{tag_p^i}(w_i)$
For i from 1 to m
w'_i
$\mathsf{Com}_{tag_i}(v_i^b)$
w'_j
(if $\mathcal{A}$ does not send w'_j, set $w'_j := 0^n$)
1, if $w'_j = v$
0, otherwise

Figure 3.4: Graphical depiction of the behavior of the adversary $\mathcal{B}_1$ in the ε-one-way security game for the commitment scheme Com. Note that $\vec{v}^0 = (v_1^0, \dots, v_m^0)$ and $\vec{v}^1 = (v_1^1, \dots, v_m^1)$.

game for the commitment scheme Com. His strategy is to mimic the experiment for $\mathcal{A}$ in the concurrent non-malleability security game and to replace a random puzzle commitment with the challenge he got from his own experiment. Note that depending on the behavior of $\mathcal{A}$, it may at some time happen that $\mathcal{A}$ sends a puzzle to who he believes is the receiver, but is actually $\mathcal{B}_1$. If $\mathcal{B}_1$ receives such a puzzle $\mathsf{Com}_{\widetilde{tag}_p^i}(\tilde{w}_i)$ from $\mathcal{A}$, he acts as an honest receiver and sends a random solution $\tilde{w}'_i$ back. The time of $\mathcal{A}$'s interaction with the "receiver" or the contents of the puzzle do not matter in this case, therefore this interaction is omitted in Fig. 3.4.

By construction, $\mathcal{B}_1$ wins the game if w'_j equals v, which happens if $\mathcal{A}$ correctly solves the j^{th} puzzle. Thus, since Com is concurrent non-malleable (and hence ϵ-one-way), there exists a negligible function negl_1 such that the following holds

$$\begin{aligned}
\mathsf{negl}_1(n) &\geq \mathsf{Pr}[\mathsf{Exp}^{\mathsf{ow}}_{\mathsf{Com},\mathcal{B}_1,\varepsilon}(n) = 1] \\
&\geq \mathsf{Pr}[\mathsf{Exp}^{\mathsf{ow}}_{\mathsf{Com},\mathcal{B}_1,\varepsilon}(n) = 1 \mid \exists i : \mathcal{A} \text{ solves puzzle } i] \\
&\quad \cdot \mathsf{Pr}[\exists i : \mathcal{A} \text{ solves puzzle } i] \\
&\geq \frac{1}{m} \cdot \mathsf{Pr}[\exists i : \mathcal{A} \text{ solves puzzle } i] \\
&\geq \frac{1}{m} \cdot \mathsf{Pr}[\mathsf{Exp}^{\mathsf{cnm}}_{\mathsf{Com}',\mathcal{A},\mathcal{D}_\mathcal{A}}(n) = 1 \land \exists i : \mathcal{A} \text{ solves puzzle } i]
\end{aligned} \tag{3.2}$$

Case 2: $\mathcal{A}$ solves none of the m puzzles. Using $\mathcal{A}$, we construct an adversary $\mathcal{B}_2$ against the concurrent non-malleability of the commitment scheme Com.

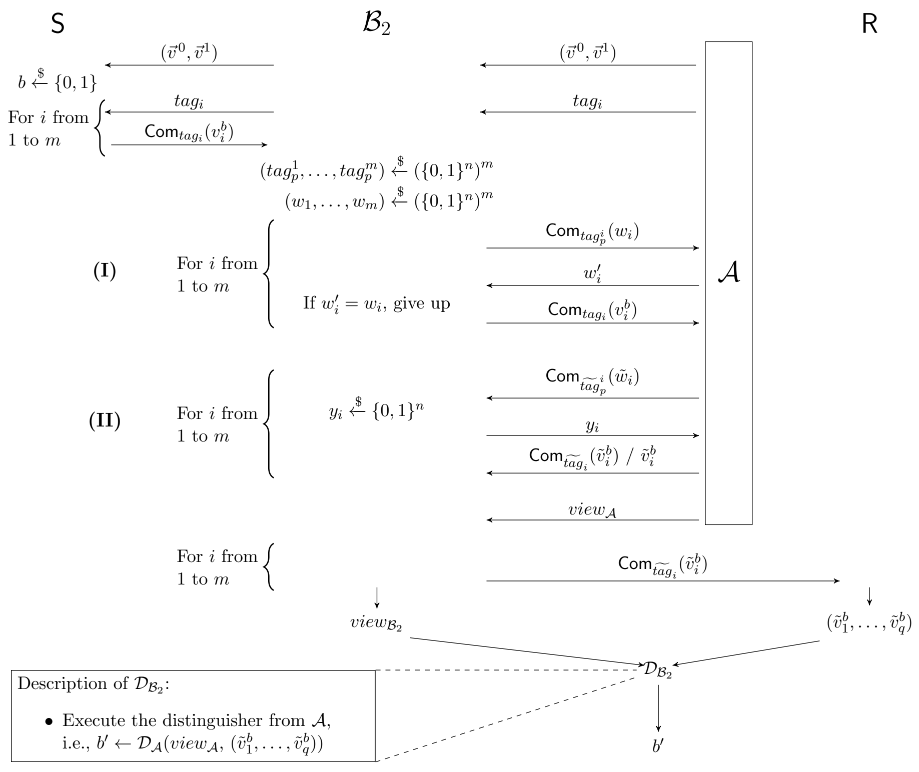

S
$\mathcal{B}_2$
R
$(\vec{v}^0, \vec{v}^1)$
$b \xleftarrow{\$} \{0,1\}$
For i from 1 to m
tag_i
$\mathsf{Com}_{tag_i}(v_i^b)$
$(tag_p^1, \ldots, tag_p^m) \xleftarrow{\$} (\{0,1\}^n)^m$
$(w_1, \ldots, w_m) \xleftarrow{\$} (\{0,1\}^n)^m$
(I)
For i from 1 to m
If $w_i' = w_i$, give up
$\mathsf{Com}_{tag_p^i}(w_i)$
w_i'
$\mathsf{Com}_{tag_i}(v_i^b)$
$\mathcal{A}$
(II)
For i from 1 to m
$y_i \xleftarrow{\$} \{0,1\}^n$
$\mathsf{Com}_{\widetilde{tag}_p^i}(\tilde{w}_i)$
y_i
$\mathsf{Com}_{\widetilde{tag}_i}(\tilde{v}_i^b)$ / $\tilde{v}_i^b$
$view_{\mathcal{A}}$
For i from 1 to m
$\mathsf{Com}_{\widetilde{tag}_i}(\tilde{v}_i^b)$
$view_{\mathcal{B}_2}$
$(\tilde{v}_1^b, \ldots, \tilde{v}_q^b)$
$\mathcal{D}_{\mathcal{B}_2}$
b'
Description of $\mathcal{D}_{\mathcal{B}_2}$:
• Execute the distinguisher from $\mathcal{A}$, i.e., $b' \leftarrow \mathcal{D}_{\mathcal{A}}(view_{\mathcal{A}}, (\tilde{v}_1^b, \ldots, \tilde{v}_q^b))$

Figure 3.5: Graphical depiction of the behavior of the adversary $\mathcal{B}_2$ in the concurrent non-malleability security game for the commitment scheme Com. At **(I)** $\mathcal{A}$'s interaction with the "sender" is depicted and at **(II)** $\mathcal{A}$'s interaction with the "receiver". Note that $\vec{v}^0 = (v_1^0, \ldots, v_m^0)$ and $\vec{v}^1 = (v_1^1, \ldots, v_m^1)$. Note that $\mathsf{Com}_{\widetilde{tag}_i}(\tilde{v}_i^b)$ / $\tilde{v}_i^b$ denotes that, depending on whether $\mathcal{B}_2$ correctly guessed the solution y_i or not, the i^{th} result value is sent as a commitment or as a plaintext value. In the (negligible) case that $\mathcal{B}_2$ correctly solves a puzzle and gets a value $\tilde{v}_i$ as plaintext, he himself commits to this value before sending the commitment to the receiver. Also note that $view_{\mathcal{B}_2}$ contains $view_{\mathcal{A}}$.

For each $i \in \{1, \ldots, m\}$, $\mathcal{B}_2$ sends an honestly generated puzzle to $\mathcal{A}$ (thereby simulating the sender), who fails to solve it, and then forwards the i^{th} commitment he gets from the sender to $\mathcal{A}$. When $\mathcal{A}$ interacts with his receiver, who is simulated by $\mathcal{B}_2$, $\mathcal{B}_2$ answers with random strings in the puzzle phases (to simulate an honest receiver) and forwards the commitments from $\mathcal{A}$ to his own receiver (cf. Fig. 3.5).

It holds that, since Com is concurrent non-malleable, there exists a negligible function negl_2 such that

$$\begin{aligned}
\frac{1}{2} + \mathsf{negl}_2(n) &\geq \Pr[\mathsf{Exp}^{\mathsf{cnm}}_{\mathsf{Com},\mathcal{B}_2,\mathcal{D}_{\mathcal{B}_2}}(n) = 1] \\
&\geq \Pr[\mathsf{Exp}^{\mathsf{cnm}}_{\mathsf{Com},\mathcal{B}_2,\mathcal{D}_{\mathcal{B}_2}}(n) = 1 \mid \nexists i : \mathcal{A} \text{ solves puzzle } i] \\
&\quad \cdot \Pr[\nexists i : \mathcal{A} \text{ solves puzzle } i] \\
&= \Pr[\mathsf{Exp}^{\mathsf{cnm}}_{\mathsf{Com}',\mathcal{A},\mathcal{D}_{\mathcal{A}}}(n) = 1 \mid \nexists i : \mathcal{A} \text{ solves puzzle } i] \\
&\quad \cdot \Pr[\nexists i : \mathcal{A} \text{ solves puzzle } i] \\
&= \Pr[\mathsf{Exp}^{\mathsf{cnm}}_{\mathsf{Com}',\mathcal{A},\mathcal{D}_{\mathcal{A}}}(n) = 1 \land \nexists i : \mathcal{A} \text{ solves puzzle } i]
\end{aligned} \tag{3.3}$$

Putting things together. Combining Eq. 3.2, Eq. 3.3 and Eq. 3.1, we get the following:

$$\begin{aligned}
\Pr[\mathsf{Exp}^{\mathsf{cnm}}_{\mathsf{Com}',\mathcal{A},\mathcal{D}_{\mathcal{A}}}(n) = 1] &= \Pr[\mathsf{Exp}^{\mathsf{cnm}}_{\mathsf{Com}',\mathcal{A},\mathcal{D}_{\mathcal{A}}}(n) = 1 \land \exists i : \mathcal{A} \text{ solves puzzle } i] \\
&\quad + \Pr[\mathsf{Exp}^{\mathsf{cnm}}_{\mathsf{Com}',\mathcal{A},\mathcal{D}_{\mathcal{A}}}(n) = 1 \land \nexists i : \mathcal{A} \text{ solves puzzle } i] \\
&\leq m \cdot \mathsf{negl}_1(n) + \mathsf{negl}_2(n) + \frac{1}{2}
\end{aligned}$$

This concludes the proof of the theorem.

□

More Separations based on the Puzzle-Solution Approach

In this section, we show how more separation results can be obtained by appropriately instantiating the puzzle-solution approach from Sec. 3.2.5.

First, using the same puzzle and very similar arguments as in the proof of Thm. 3.17, one can also prove that parallel non-malleability does not imply parallel CCA security, that (stand-alone) non-malleability does not imply one-one CCA security, that concurrent non-malleability does not imply parallel CCA security and that parallel non-malleability does not imply one-one CCA security.

Theorem 3.18 (PNM $\not\Rightarrow$ PCCA)**.** *If* Com *is a statistically binding, parallel non-malleable commitment scheme, then* $\mathsf{Com}' \leftarrow \mathsf{PComGen}(\mathsf{Com}, 1, seq)$ *is also statistically binding and parallel non-malleable but not parallel CCA-secure.*

Theorem 3.19 (NM $\not\Rightarrow$ 1CCA)**.** *If* Com *is a statistically binding, non-malleable commitment scheme, then* $\mathsf{Com}' \leftarrow \mathsf{PComGen}(\mathsf{Com}, 1, seq)$ *is also statistically binding and non-malleable but not one-one CCA-secure.*

Theorem 3.20 (CNM $\not\Rightarrow$ PCCA)**.** *If* Com *is a statistically binding, concurrent non-malleable commitment scheme, then* $\mathsf{Com}' \leftarrow \mathsf{PComGen}(\mathsf{Com}, 1, seq)$ *is also statistically binding and concurrent non-malleable but not parallel CCA-secure.*

Theorem 3.21 (PNM $\not\Rightarrow$ 1CCA)**.** *If* Com *is a statistically binding, parallel non-malleable commitment scheme, then* $\mathsf{Com}' \leftarrow \mathsf{PComGen}(\mathsf{Com}, 1, seq)$ *is also statistically binding and parallel non-malleable but not one-one CCA-secure.*

One can prove additional separations using other puzzles.

Theorem 3.22 (1CCA $\not\Rightarrow$ PCCA)**.** *If* Com *is a statistically binding, one-one CCA-secure commitment scheme, then* $\mathsf{Com}' \leftarrow \mathsf{PComGen}(\mathsf{Com}, 2, par)$ *is also statistically binding and one-one CCA-secure but not parallel CCA-secure.*

Proof idea. The puzzle consists of two parallel commitments. It is thus solvable with a parallel CCA-oracle but not with a one-one CCA-oracle. The probability that in the reduction of the first case of the second claim the oracle query can be answered is at least $1/2 - 1/2^n$ (where n is the tag length).

$\square$

Theorem 3.23 (PCCA $\not\Rightarrow$ CCA)**.** *If* Com *is a statistically binding, parallel CCA-secure commitment scheme, then* $\mathsf{Com}' \leftarrow \mathsf{PComGen}(\mathsf{Com}, 2, seq)$ *is also statistically binding and parallel CCA-secure but not CCA-secure.*

Proof idea. The puzzle consists of two commitments sent sequentially. It is thus solvable with a CCA-oracle but not with a parallel CCA-oracle. The probability that in the reduction of the first case of the second claim the oracle query can be answered is at least $1/2 - m/2^n$ (where m is the number of commitments in the oracle query and n the tag length).

$\square$

3.2.6 Second Transformation (Sharing Approach)

In this section, we settle the remaining separations. Up to now we have been able to prove our separations using the puzzle-solution approach. However, in order to prove the remaining separations, we cannot use this approach anymore. This is because we need to construct commitment schemes that do not fulfill a certain *variant of non-malleability* for the remaining separations. We can therefore no longer insert a puzzle into a given commitment scheme since an adversary in a non-malleability-related experiment does not have a committed value oracle at his disposal that can be used to solve the puzzle.

We therefore deviate from the puzzle-solution approach in the following way: Instead of sending a puzzle, i.e., commitments to random strings, we let the sender commit to *shares* of the message to be committed to using two different random tags. This way, the adversary will be able to forward the commitments to the shares to the receiver in his experiment. After the commit phase is over, these shares will then be opened by the implicit oracle in the experiment. The distinguisher will then be able to reconstruct the message and win in the experiment.

Using the above approach, we first show that parallel CCA security does not imply concurrent non-malleability. To this end, consider the following scheme Com', given a commitment scheme Com:

On input $v \in \{0,1\}^n$, $tag \in \{0,1\}^n$, the sender generates shares $s_0, s_1 \in \{0,1\}^n$ such that $s_0 \oplus s_1 = v$. He then sends $\mathsf{Com}_{tag_0}(s_0)$ and $\mathsf{Com}_{tag_1}(s_1)$ to the receiver in a *sequential* order using random tags $tag_0, tag_1 \leftarrow \{0,1\}^n$. Afterwards, the sender sends $\mathsf{Com}_{tag}(v)$ to the receiver. The unveil phase is the same as in Com (note that the shares are never unveiled).

First note that, in general, the above construction Com' does not yield a separation between concurrent non-malleability and parallel CCA security, even if Com is parallel CCA-secure. This is because Com' may fulfill *neither* of these two security notions. For instance, assuming Com is *non-interactive*, an adversary against the parallel CCA security of Com' can simply forward the two commitments to the shares to his oracle and thereby easily win in his experiment.

In order to obtain a separation, we therefore additionally assume that Com is *extractable* (cf. Definition 2.10). Note that if a statistically binding, parallel

CCA-secure commitment scheme exists, then there also exists a statistically binding, parallel CCA-secure commitment scheme that is additionally extractable. This is follows from the fact that one-way functions can be constructed from commitment schemes (in a black-box way) [IL89] and that extractable CCA-secure commitment schemes can be constructed from one-way functions (in a black-box way) [Kiy14].

In the proof of the separation between concurrent non-malleability and parallel CCA security, we use the following auxiliary experiment:

Definition 3.24 (RepeatPCCA)**.** *RepeatPCCA is like the ordinary parallel CCA security game except that the adversary can "reset" the experiment at any given moment.*

More specifically, the adversary (on input 1^n, z) first chooses two strings (v_0, v_1) and a challenge tag tag *and sends (v_0, v_1, tag) to the experiment. The experiment then chooses a random bit $b \leftarrow \{0,1\}$ and commits to v_b using the tag* tag*. The adversary can then send* `reset` *to the experiment or a bit b'. If the adversary sends* `reset`*, then he can send new strings (v_0', v_1') and a new challenge tag to the experiment. The experiment then commits to v_b' using the new challenge tag (note that the challenge bit b remains the same.) The adversary may reset the experiment polynomially many times. If the adversary sends a bit b', then the experiment outputs 1 if $b = b'$ and 0 otherwise. Throughout the experiment, the adversary may send a single parallel query to $\mathcal{O}_{\mathsf{pcca}}$. If the adversary sends* `reset` *but hasn't finished his query yet, then his query is invalidated, i.e., the oracle ignores all further messages.*

An adversary $\mathcal{A}$ is called valid *if he only chooses strings v_0, v_1 such that $|v_0| = |v_1|$ and does not query the parallel-CCA oracle on tags that are equal to the* current *challenge tag during the experiment.*

Denote by $\mathsf{Exp}^{\mathsf{rpcca}}_{\mathsf{Com},\mathcal{A}(z)}(n)$ the output of the above experiment. We say that a tag-based commitment scheme Com *is* RepeatPCCA-secure *if for every valid PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that for all $n \in \mathbb{N}, z \in \{0,1\}^*$ it holds that*

$$\Pr[\,\mathsf{Exp}^{\mathsf{rpcca}}_{\mathsf{Com},\mathcal{A}(z)}(n) = 1\,] \leq \frac{1}{2} + \mathsf{negl}(n)$$

We have the following lemma:

Lemma 3.25. *If a commitment scheme is parallel CCA-secure and extractable, then it is also RepeatPCCA-secure.*

Proof idea. Denote by (v_0^i, v_1^i, tag_i) the challenge strings (v_0^i, v_1^i) and the challenge tag tag_i the adversary $\mathcal{A}$ sends in the i^{th} session. Using a standard hybrid argument, one can replace all challenge commitments $\mathsf{Com}(v_b^i, tag_i)$ with commitments $\mathsf{Com}(0^{|v_0^i|}, tag_i)$ to all-0 strings. Indistinguishability follows by reduction to parallel CCA security. Note that the reduction $\mathcal{B}$ can answer the oracle query of the adversary $\mathcal{A}$ against the RepeatPCCA security in the following way: If $\mathcal{A}$ sends his query during $\mathcal{B}$'s challenge phase, then $\mathcal{B}$ forwards the query to his own parallel CCA-oracle. If $\mathcal{A}$ sends his query before or after $\mathcal{B}$'s challenge phase, then $\mathcal{B}$ uses the extractability property.

□

We are now ready to prove the following theorem:

Theorem 3.26 (PCCA $\not\Rightarrow$ CNM)**.** *If there exists a statistically binding, parallel CCA-secure commitment scheme, then there also exists a statistically binding and parallel CCA-secure commitment scheme that is not concurrent non-malleable.*

Proof. Let Com' be as above with a statistically binding, parallel CCA-secure and extractable commitment scheme Com as its base commitment scheme (as noted above, such a Com exists if a statistically binding, parallel CCA-secure commitment scheme exists).

The statistical binding property of Com' follows readily from the statistical binding property of the underlying commitment scheme Com. In the following, we prove that Com' is parallel CCA-secure but not concurrent non-malleable.[10]

Claim 1: Com' is not concurrent non-malleable. A man-in-the-middle adversary in the concurrent non-malleability game sends $\big((v_1^0, \ldots, v_m^0), (v_1^1, \ldots, v_m^1)\big)$ and tags $tag_1, \ldots, tag_m$ to the sender, who randomly selects a bit b. The sender then commits for each $i \in \{1, \ldots, m\}$ to the shares $s_{i_0}^b$ and $s_{i_1}^b$ using random tags and to v_i^b using tag tag_i to the adversary (with $s_{i_0}^b \oplus s_{i_1}^b = v_i^b$). Let $h := \lfloor \frac{m}{2} \rfloor$. For each $j \in \{1, \ldots, h\}$ the adversary forwards the commitments to $s_{j_0}^b$ and $s_{j_1}^b$ to the receiver (as shares for these commitments he just uses commitments to 0^n). If m is odd, he chooses 0^n as his last message to commit to (he also uses commitments to 0^n as shares). Since the random tags that are used to commit to the shares (and possibly 0^n) are not in $\{tag_1, \ldots, tag_m\}$ with overwhelming probability, the distinguisher is given the shares $(s_{1_0}^b, s_{1_1}^b, \ldots, s_{h_0}^b, s_{h_1}^b)$ as input (and possibly 0^n) and can thus reconstruct $(v_1^b, \ldots, v_h^b)$, which suffices to deduce the correct b if the challenge messages are chosen appropriately.

Claim 2: Com' is parallel CCA-secure. Let $\mathcal{A}$ be a PPT adversary against the parallel CCA security of Com'. Consider the following hybrids for the commitment scheme Com': H_0 is the ordinary parallel CCA security game. H_1 is like H_0, except that the sender now commits to two random and *independently distributed* strings s_0, t (that therefore do not fulfill $s_0 \oplus t = v$ in general). Finally, H_2 is like H_1, except that the sender commits to 0^n instead of (his input) v.

Let out_i be the output of the hybrid H_i.

Sub-Claim 1: $|\mathsf{Pr}[out_0 = 1] - \mathsf{Pr}[out_1 = 1]| \leq \mathsf{negl}(n)$. Consider the following adversary $\mathcal{B}$ against Com in the RepeatPCCA game: The adversary $\mathcal{B}$ simulates the experiment H_0 for $\mathcal{A}$. $(*)$ After $\mathcal{A}$ has sent (v_0, v_1, tag), $\mathcal{B}$ chooses a random bit $b \leftarrow \{0,1\}$ and generates shares s_0, s_1 such that $s_0 \oplus s_1 = v_b$ and a random string $t \in \{0,1\}^n$. The adversary $\mathcal{B}$ then sends (s_1, t, tag_1), where tag_1 is a random tag of length n, to his experiment. Afterwards, $\mathcal{B}$ randomly selects one of the two (sequentially ordered) commit sessions to the shares of v_b in the commit phase of Com' and inserts his challenge C* into the selected session and $\mathsf{Com}_{tag_0}(s_0)$ into the other session (for a randomly chosen tag $tag_0 \in \{0,1\}^n$). If the adversary $\mathcal{A}$ starts his (parallel) oracle query *during the challenge phase of* $\mathcal{B}$ (i.e., during the session in which $\mathcal{B}$ has inserted his challenge C*), then $\mathcal{B}$

[10]For ease of notation, we again omit the (non-uniform) input z of the adversary and distinguisher. The proof can be easily adapted to include this input.

resets his experiment and repeats the aforementioned strategy (i.e., jumps back to $(*)$). Otherwise, $\mathcal{B}$ answers $\mathcal{A}$'s oracle query in the following way:

Case 1: If $\mathcal{A}$ starts his query *before* $\mathcal{B}$'s challenge phase has begun *and* $\mathcal{A}$'s query does not use $\mathcal{B}$'s challenge tag tag_1, then $\mathcal{B}$ forwards $\mathcal{A}$'s query to his own parallel CCA-oracle (if $\mathcal{A}$'s query uses $\mathcal{B}$'s challenge tag, then $\mathcal{B}$ aborts).

Case 2: If $\mathcal{A}$ starts his query *after* $\mathcal{B}$'s challenge phase is over, then $\mathcal{B}$ answers the query by extracting $\mathcal{A}$.[11]

Afterwards, $\mathcal{B}$ continues simulating the experiment H_0 for $\mathcal{A}$. After the simulated experiment is over, $\mathcal{B}$ outputs what the simulated experiment outputs. The adversary $\mathcal{B}$ repeats the experiment at most $n-1$ times (and aborts if the n^{th} iteration leads to another reset).

Denote by $\mathbf{E}_{\text{badquery}}$ the event that the adversary $\mathcal{A}$ queries the parallel CCA-oracle during the challenge phase of $\mathcal{B}$ in *all* iterations.

Let $j \in \{1,2\}$ be the session into which $\mathcal{B}$ has chosen to insert his challenge C^*. Since $\mathcal{B}$ chooses j *randomly* in each iteration and $\mathcal{A}$'s view is *independent* of j in each iteration, it holds that $\mathsf{Pr}[\mathbf{E}_{\text{badquery}}] \leq 1/2^n$.

Denote by $\mathbf{E}_{\text{guesstag}}$ the event that $\mathcal{A}$ queries his parallel CCA-oracle *before* the challenge C^* has started *using $\mathcal{B}$'s challenge tag tag_1* in one of the iterations.

Since the challenge tag tag_1 is chosen randomly (from the set of strings of length n) and $\mathcal{A}$'s view is independent of tag_1 before the challenge phase C^* begins, it holds that $\mathsf{Pr}[\mathbf{E}_{\text{guesstag}}] \leq n \cdot i/2^n$, where $i = poly(n)$ is the number of commitments in the parallel oracle query.

Now it holds that conditioned on $\mathbf{E}_{\text{badquery}}$ and $\mathbf{E}_{\text{guesstag}}$ both *not* occurring, the output of $\mathcal{B}$ is either identically distributed to the output of H_0 (this holds if $\mathrm{C}^* = \mathsf{Com}_{tag_1}(s_1)$) or identically distributed to the output of H_1 (this holds if $\mathrm{C}^* = \mathsf{Com}_{tag_1}(t)$).

Let $E = \mathbf{E}_{\text{badquery}} \cup \mathbf{E}_{\text{guesstag}}$ and let $\text{Output}^{\mathsf{rpcca}}_{\mathsf{Com},\mathcal{B}}(b)$ denote the output of $\mathcal{B}$ in the RepeatPCCA-experiment if the challenge bit b was chosen by the RepeatPCCA-experiment. Then we have the following:

$$\begin{aligned}
|\mathsf{Pr}[out_0 = 1] - \mathsf{Pr}[out_1 = 1]| &\leq \mathsf{Pr}[E] + |\mathsf{Pr}[out_0 = 1|\neg E] - \mathsf{Pr}[out_1 = 1|\neg E]| \\
&= \mathsf{Pr}[E] + |\mathsf{Pr}[\text{Output}^{\mathsf{rpcca}}_{\mathsf{Com},\mathcal{B}}(0) = 1|\neg E] \\
&\quad - \mathsf{Pr}[\text{Output}^{\mathsf{rpcca}}_{\mathsf{Com},\mathcal{B}}(1) = 1|\neg E]| \\
&\leq \frac{n \cdot i + 1}{2^n} + \mathsf{negl}(n) \\
&= \mathsf{negl}'(n)
\end{aligned}$$

Note that $|\text{Output}^{\mathsf{rpcca}}_{\mathsf{Com},\mathcal{B}}(0) = 1|\neg E] - \mathsf{Pr}[\text{Output}^{\mathsf{rpcca}}_{\mathsf{Com},\mathcal{B}}(1) = 1|\neg E]| \leq \mathsf{negl}(n)$ holds because Com is RepeatPCCA-secure by Lemma 3.25 and $\mathsf{Pr}[\neg E] = 1 - (n \cdot i+1)/2^n$ is overwhelming in n (see technical remark).

[11]Note that, in general, $\mathcal{B}$ cannot use his own oracle in case 2. This is because, in this case, $\mathcal{A}$ queries his parallel CCA-oracle after $\mathcal{B}$'s challenge phase is over. Hence, $\mathcal{A}$ knows the challenge tag tag_1 and may query his parallel CCA-oracle using tag_1. Therefore, $\mathcal{B}$ cannot simply forward $\mathcal{A}$'s query to his own parallel CCA-oracle since $\mathcal{A}$'s query may contain $\mathcal{B}$'s challenge tag. Furthermore, $\mathcal{B}$ cannot use the extractability property in case 1 since the messages of $\mathcal{A}$'s oracle query and the messages of $\mathcal{B}$'s challenge phase may *overlap* in this case. Hence, $\mathcal{B}$ cannot extract $\mathcal{A}$ since this may require "rewinding" the experiment of $\mathcal{B}$ to a specific point in $\mathcal{B}$'s challenge phase.

Technical Remark:
Let $\mathrm{Output}(b) = \mathrm{Output}^{\mathsf{rpcca}}_{\mathsf{Com},\mathcal{B}}(b)$

$$|\mathsf{Pr}[\mathrm{Output}(0) = 1] - \mathsf{Pr}[\mathrm{Output}(1) = 1]| =$$

$$|(\mathsf{Pr}[\mathrm{Output}(0) = 1 \wedge \neg E] - \mathsf{Pr}[\mathrm{Output}(1) = 1 \wedge \neg E]) + (\mathsf{Pr}[\mathrm{Output}(0) = 1 \wedge E] - \mathrm{Output}(1) = 1 \wedge E])|$$

$$\geq |\mathsf{Pr}[\mathrm{Output}(0) = 1 \wedge \neg E] - \mathrm{Output}(1) = 1 \wedge \neg E]| - |\mathsf{Pr}[\mathrm{Output}(0) = 1 \wedge E] - \mathrm{Output}(1) = 1 \wedge E]|$$
(because $|x + y| \geq |x| - |y|$)

$$\geq \mathsf{Pr}[\neg E] \cdot |\mathsf{Pr}[\mathrm{Output}(0) = 1|\neg E] - \mathsf{Pr}[\mathrm{Output}(1) = 1|\neg E]| - \mathsf{Pr}[E]$$
(because $|\mathsf{Pr}[\mathrm{Output}(0) = 1 \wedge E] - \mathrm{Output}(1) = 1 \wedge E]| \leq \mathsf{Pr}[E]$)

$$\geq \frac{1}{2} \cdot |\mathsf{Pr}[\mathrm{Output}(0) = 1|\neg E] - \mathsf{Pr}[\mathrm{Output}(1) = 1|\neg E]| - \mathsf{Pr}[E]$$
(This holds for sufficiently large n because $\mathsf{Pr}[\neg E]$ is overwhelming. Note that $1/2$ is arbitrary, any constant $0 < c < 1$ works)

Since Com is RepeatPCCA-secure, $|\mathrm{Output}(0) = 1] - \mathsf{Pr}[\mathrm{Output}(1) = 1]|$ is negligible.
Furthermore, $\mathsf{Pr}[E] = \frac{n \cdot i + 1}{2^n}$ is negligible. Hence, $|\mathsf{Pr}[\mathrm{Output}(0) = 1|\neg E] - \mathsf{Pr}[\mathrm{Output}(1) = 1|\neg E]|$ must also be negligible.

Sub-Claim 2: $|\mathsf{Pr}[out_1 = 1] - \mathsf{Pr}[out_2 = 1]| \leq \mathsf{negl}(n)$. This follows from a standard reduction argument to the parallel CCA security of Com. Consider an adversary $\mathcal{B}'$ against the parallel CCA security of Com. The adversary $\mathcal{B}'$ simulates the experiment H_1 for $\mathcal{A}$. After $\mathcal{A}$ has sent (v_0, v_1, tag), $\mathcal{B}'$ chooses a random bit $b \leftarrow \{0, 1\}$ and sends $(v_b, 0^n, tag)$ to his experiment. Afterwards, $\mathcal{B}'$ forwards his challenge C^* to $\mathcal{A}$ as $\mathcal{A}$'s challenge. If $\mathcal{A}$ queries his oracle, then $\mathcal{B}'$ forwards this query to his own oracle. After the simulated experiment is over, $\mathcal{B}'$ outputs what the simulated experiment outputs. It holds that the output of $\mathcal{B}'$ is identically distributed to the output of H_1 if $\mathrm{C}^* = \mathsf{Com}_{tag}(v_b)$ and identically distributed to the output of H_2 if $\mathrm{C}^* = \mathsf{Com}_{tag}(0^n)$. Sub-Claim 2 now follows from the parallel CCA security of Com.

Sub-Claim 3: $\mathsf{Pr}[out_2 = 1] = 1/2$. This follows from the fact that the view of $\mathcal{A}$ in the hybrid H_2 is independent of the challenge bit.

In conclusion, $|\mathsf{Pr}[out_0 = 1] - 1/2]| \leq \mathsf{negl}(n)$. Hence, Com' is parallel CCA-secure.
□

Using the transformation implied by [IL89; Kiy14] described earlier, Thm. 3.26 and the fact that parallel CCA security implies parallel non-malleability, we also get the following separation:

Theorem 3.27 (PNM $\not\Rightarrow$ CNM)**.** *If there exists a statistically binding, parallel non-malleable commitment scheme, then there also exists a statistically binding and parallel non-malleable commitment scheme that is not concurrent non-malleable.*

Using similar arguments as in the proof of Thm. 3.26, one can also show that one-one CCA security does not imply parallel non-malleability.

Theorem 3.28 (1CCA $\not\Rightarrow$ PNM)**.** *If there exists a statistically binding, one-one CCA-secure commitment scheme, then there also exists a statistically binding and one-one CCA-secure commitment scheme that is not parallel non-malleable.*

Proof idea. This separation follows by adapting the techniques used for the separation in Thm. 3.26. In the commitment scheme Com' the sender commits to the shares s_0 and s_1 *in parallel* instead of sequentially. The experiment Repeat1CCA is like RepeatPCCA except that the adversary may now query $\mathcal{O}_{\mathsf{1cca}}$ instead of $\mathcal{O}_{\mathsf{pcca}}$.

□

Remark 3.29. *We note that the (known) separation between (stand-alone) non-malleability and parallel non-malleability can also be proven using the sharing approach. This follows from the transformation implied by [IL89; Kiy14], Thm. 3.28 and the fact that one-one CCA security implies (stand-alone) non-malleability.*

Remark 3.30. *We remark that all results, except for Thms. 3.19 and 3.21, carry over to* bit *commitment schemes. This can be shown by similar arguments as in the proofs of Thms. 3.17 and 3.26. The main difference for the proofs using the puzzle-solution approach is that the puzzle consists of n parallel (bit) commitments. The main difference for the proofs using the sharing approach is that the sender generates $2n$ shares. We do not know if Thms. 3.19 and 3.21 carry over to bit commitment schemes because those theorems cannot be proven using the above modification of the puzzle-solution approach. This is because the number of queries that can be sent to the oracle in these cases is bounded by a constant. Hence, the oracle cannot be used to solve a puzzle consisting of n parallel bit commitments.*

Remark 3.31. *Note that if one-way functions exist, then all base commitment schemes required for this chapter exist. In all results one can use, e.g., the commitment scheme from [CLP10] that is based on one-way functions as base commitment scheme* Com*. This scheme is CCA-secure and therefore fulfills every security notion considered in this chapter.*

Chapter 4

A New Framework for Concurrently Composable MPC in the Plain Model

4.1 Introduction

In this chapter, we present a new framework for concurrently composable MPC in the plain model. Before presenting our framework and its properties, we first recall the impossibility results of the UC framework and how these have been overcome in the literature so far.

Impossibility Results. It has been proven in the literature that many interesting functionalities such as commitments, zero-knowledge or oblivious transfer cannot be UC-realized in the plain model (see, e.g. [CF01; CKL03; Lin03; PR08; KL11])). Furthermore, [Lin04] proved that the need for some additional trusted setup extends to even the special case of (concurrent) *self*-composability, where only instances of the *same* protocol are concurrently executed. As an example of these negative results, we recall the argument in [CF01] for the impossibility of UC-realizing commitments in the plain model (we have slightly adapted the argument to better fit the discussion that comes after it). Impossibility of UC-realizing other functionalities follow from conceptually similar arguments.

Consider an environment $\mathcal{Z}$ interacting with the dummy adversary $\mathcal{D}$ and a commitment protocol π in the plain model. At the beginning, $\mathcal{Z}$ instructs $\mathcal{D}$ to corrupt the sender. $\mathcal{Z}$ then randomly chooses a bit b and commits to b by running the program of the honest sender and interacting with the honest receiver via $\mathcal{D}$. In the unveil phase, $\mathcal{Z}$ unveils its committed value. At the end of the protocol execution, $\mathcal{Z}$ checks if the bit b' output by the honest receiver equals the b that $\mathcal{Z}$ chose at the beginning. $\mathcal{Z}$ outputs 1 if and only if $b' = b$.

Assume for the sake of contradiction that π UC-realizes $\mathcal{F}_{\text{com}}$. Then there exists a simulator $\mathcal{S}$ such that $\text{Exec}_{\mathsf{UC}}(\pi, \mathcal{D}, \mathcal{Z}) \overset{c}{\equiv} \text{Exec}_{\mathsf{UC}}(\mathcal{F}_{\text{com}}, \mathcal{S}, \mathcal{Z})$. $\mathcal{S}$ must be able to extract the bit b from the protocol transcript with overwhelming probability in order to input b into $\mathcal{F}_{\text{com}}$ in the commit phase. One can now construct an environment that uses $\mathcal{S}$ to break the protocol π. Consider the environment $\mathcal{Z}_{\mathcal{S}}$ interacting with the dummy adversary $\mathcal{D}$ and the protocol π. At the beginning, $\mathcal{Z}_{\mathcal{S}}$ instructs $\mathcal{D}$ to corrupt the receiver. $\mathcal{Z}_{\mathcal{S}}$ then chooses a random bit b and hands b to the honest sender as input. $\mathcal{Z}_{\mathcal{S}}$ internally runs the simulator $\mathcal{S}$, relaying the messages between the honest sender and $\mathcal{S}$. When $\mathcal{S}$ outputs a bit b' as input for $\mathcal{F}_{\text{com}}$ in $\mathcal{Z}_{\mathcal{S}}$'s internal simulation, $\mathcal{Z}_{\mathcal{S}}$ checks if $b' = b$ and outputs 1 if this holds and 0 otherwise. If $\mathcal{S}$ does not output anything, $\mathcal{Z}_{\mathcal{S}}$ outputs a random bit.

It holds that $\mathcal{S}$ outputs the bit b with overwhelming probability when $\mathcal{Z}_{\mathcal{S}}$ interacts with the real protocol. However, since the view of an ideal-model adversary is statistically independent of the bit b, $\mathcal{S}$ outputs the bit b with probabilty at most 1/2 when $\mathcal{Z}_{\mathcal{S}}$ interacts with the ideal protocol and any ideal-model adversary. $\mathcal{Z}_{\mathcal{S}}$ is therefore able to distinguish between the real and ideal protocol with non-negligible probability. We have thus reached a contradiction.

Previous Approaches to Overcome Impossibility Results. The foregoing argument goes through because the environment is be able to internally run the simulator $\mathcal{S}$ in such a way that it can use $\mathcal{S}$ to break the protocol π. In order to overcome this impossibility result, one must therefore find a way to prevent the environment from exploiting the capabilities of the simulator.

One solution to this problem is to rely on some additional trusted setup assumptions. Rather than designing a protocol in the plain model like π in the above argument, one works in the $\mathcal{F}$-hybrid model for some functionality $\mathcal{F}$ modeling a trusted setup. Since the environment does not have direct access to $\mathcal{F}$ but only via the adversary, the simulator is able to report a "rigged" setup to the environment that allows him to simulate successfully. For instance, if $\mathcal{F}$ is some form of public key infrastructure, the simulator is able to report public keys to which he knows the corresponding secret keys, simply by generating all keys by himself. Since in the real protocol the setup is honestly constructed by $\mathcal{F}$, the environment is not able use the simulator to break the protocol. Numerous UC-secure general MPC protocols have been constructed based on this approach (e.g. [Can+02; Bar+04; Can+07; KLP07; Kat07; CPS07; LPV09; LPV12; Dac+13a; HV15]). For instance, [Can+02] provided a construction based on a *common reference string* where the trusted setup is assumed to provide a string that is honestly sampled from some predefined distribution. [Bar+04] showed how to achieve general MPC in the UC framework assuming various forms of *public key infrastructures* where the key registration authority knows the secrets keys to the registered public keys. [Kat07] constructed a protocol based on *tamper-proof hardware tokens* as setup assumption. However, protocols which assume some additional trusted setup are less likely to be secure in practice, where a trusted setup is often hard to come by (or expensive). It is therefore important to have a meaningful composable security notion that is achievable in the plain model.

In order to reach this goal, new frameworks relaxing UC security have been proposed in the literature. One of the most prominent approaches in this line of research is based on the idea of granting the simulator *super-polynomial resources*. This approach is motivated by the fact that many ideal functionalities such as commitments or oblivious transfer are information-theoretically secure, i.e. their security is independent of the runtime of the ideal-model adversary.[1]

[1]For instance, the (bit) commitment functionality $\mathcal{F}_{\text{com}}$ (cf. Definition 2.25) only outputs a notification message after receiving an input from the sender. Therefore, the hiding property

[Pas03] initiated this approach by introducing the *Security with super-polynomial simulators* (SPS) framework where the simulator may run in super-polynomial time while the environment and adversary are PPT. Many self-composable general MPC protocols in the plain model have been constructed in this framework (e.g. [Pas03; BS05; LPV12; Gar+12; Dac+13a; GKP17; GKP18]). However, allowing the simulator to perform arbitrary super-polynomial computations (while the environment remains PPT) leads to a security notion that is not closed under protocol composition. SPS security does therefore not have the advantages provided by the UC composition theorem, i.e. concurrent security and modular analysis (cf. Section 2.4.5).

To overcome this issue, [PS04] proposed the *Angel-based security framework*. In this framework, the adversary, environment and simulator are PPT but all have access to an oracle called "*Imaginary angel*" that grants super-polynomial resources for *specific* computational problems. It is easy to see that Angel-based security implies SPS security. In addition, like UC security, Angel-based security is closed under protocol composition. Furthermore, the impossibility results of the UC framework are circumvented by allowing the Imaginary Angel to base its answers on the set of corrupted parties. This ensures that the environment is unable to internally run the simulator in such a way that it can break the protocol. In particular, the above argument for the impossibility of UC-realizing commitments does not go through anymore. This is because the Imaginary Angel behaves differently in the interaction with the simulator $\mathcal{S}$, where the sender is corrupted, than in the interaction with the environment $\mathcal{Z}_{\mathcal{S}}$, where the receiver is corrupted. $\mathcal{Z}_{\mathcal{S}}$ is therefore unable to correctly answer $\mathcal{S}'s$ oracle queries in its internal simulation. [CLP10] later proposed the *UC with super-polynomial helpers* framework which also provides a composable security notion implying SPS security. In this framework, the environment and simulator may query a "helper" that, unlike Imaginary Angels, may be interactive and stateful. Many general MPC protocols in the plain model have been constructed in the Angel-based security and UC with super-polynomial helpers framework. However, all known general MPC protocols in these frameworks are either based non-standard or super-polynomial-time assumptions [PS04; MMY06; KMO14] or require a super-constant number of rounds [CLP10; LP12; Kiy14; Goy+15; HV16].

Our New Framework: "Shielded Oracles". We present a new framework that is also based on the idea of providing simulators with super-polynomial resources. However, unlike in previous frameworks, simulators in our new framework are only given *restricted access* to the results computed in super-polynomial time. We model the super-polynomial resources as stateful oracles that are "glued" to an ideal functionality. These oracles may directly interact with the functionality without the simulator observing the communication. The outputs of these oracles may depend on the session ID of the protocol as well as the set of corrupted parties. We call these oracles *shielded oracles*.

In order to obtain a composable security notion, environments in our framework may invoke additional ideal protocols that include shielded oracles. With these *augmented environments* we are able to prove a general composition theo-

holds regardless of the runtime of the adversary. Furthermore, since $\mathcal{F}_{\text{com}}$ always outputs the bit it stored in the commit phase after receiving an unveil instruction, the binding property is als trivially ensured to hold against unbounded adversaries.

rem, which, like the UC composition theorem, guarantees concurrent security. While modular analysis is not directly implied by our composition theorem for technical reasons, one can achieve modular analysis by constructing protocols with "strong modular composition properties". Protocols with such a property can be "plugged" into large classes of UC-secure protocols in such a way that the composed protocol is secure in our framework. As a proof of concept, we present a constant-round commitment scheme with such a property.

Our new security notion can be shown to lie strictly between SPS security and Angel-based security/UC security with super-polynomial helpers. Furthermore, since all super-polynomial computations are hidden away, augmented environments do not "hurt" protocols proven secure in the UC framework. Therefore, our notion is fully compatible with UC security, i.e., UC-secure protocols are also secure in our framework. Moreover, our concept of shielding away super-polynomial resources allows us to apply a new proof technique where entities involving super-polynomial resources can be replaced by indistinguishable polynomially bounded entities. This allows us to construct protocols in the plain model using weaker primitives than in the constructions in the Angel-based security and UC with super-polynomial helpers framework. In particular, we only require parallel CCA-secure commitments schemes instead of the stronger[2] primitive of CCA-secure commitment schemes. As a consequence, we are able to prove that constant-round (black-box) general MPC in the plain model based on standard polynomial-time assumptions is feasible in our framework.

This chapter is taken almost entirely from [Bro+17], which was published at EUROCRYPT 2017.

4.1.1 Contribution

We propose a new framework that is based on the idea of granting simulators only restricted access to the results of a super-polynomial oracle. In the following, we list the results of this chapter.

New Composable Security Notion

Our new framework provides a security notion with the following properties:

- Closed under protocol composition (Theorem 4.11, Corollary 4.13)
- Implies SPS security (Proposition 4.10) and is strictly weaker than Angel-based security (Theorem 4.21)
- Compatible with UC security, i.e., protocols proven secure in the UC framework are also secure in our new framework. (Theorem 4.15, Corollary 4.16)

Commitment Scheme with Modular Composition Property

As a proof of concept, we present a constant-round commitment scheme in the plain model based on OWPs that is secure in our framework (Theorem 4.26, Corollary 4.28) and can be "plugged" into a large class of UC-secure protocols, such that the composite protocol is secure in our framework (Theorem 4.34, Corollary 4.38a). Furthermore, this construction can be made fully black-box based on verifiable perfectly binding homomorphic commitment schemes (Corollary 4.29, Corollary 4.38b).

[2]Cf. Theorem 3.23.

Constant-round (Black-box) MPC in the Plain Model

We show that constant-round general MPC in the plain model based on standard polynomial-time hardness assumptions is feasible in our framework. We present two constructions:

- A non-black-box protocol based on enhanced trapdoor permutations (Theorem 4.39a)
- A black-box construction based on verifiable perfectly binding homomorphic commitment schemes and IND-CPA-secure PKE schemes with oblivious public-key generation. To the best of our knowledge, this was the first black-box constant-round general MPC protocol satisfying a meaningful composable security notion in the plain model based on standard polynomial-time hardness assumptions.[3] (Theorem 4.39b)

In addition, we also construct constant-round zero-knowledge protocols in the plain model based on weaker assumptions than the above general MPC protocols: A non-black-box construction based on OWPs (Corollary 4.30) and a black-box construction based on verifiable perfectly binding homomorphic commitment schemes (Corollary 4.31).

Building on Parallel CCA-secure Commitments

Our constructions require weaker primitives than the general MPC protocols in the Angel-based security and UC with super-polynomial helpers framework. In particular, it suffices to use *parallel* CCA-secure commitment schemes as a building block in our constructions instead of CCA-secure commitment schemes (Theorem 4.26, Theorem 4.34). Unlike CCA-secure commitment schemes, parallel CCA-secure commitment schemes have known constant-round (black-box) instantiations based on standard polynomial-time assumptions.

We note that all the above-mentioned constructions are proven secure in the *static* corruption model.

4.1.2 Related Work

The frameworks most related to ours are the SPS security framework, the Angel-based security framework and the UC with super-polynomial helpers framework.

The SPS security framework, introduced by [Pas03], provides a meaningful security notion for many cryptographic tasks such as commitment schemes or oblivious transfer. However, unlike UC security, SPS security is not closed under protocol composition. Many general MPC protocols in the plain model have been constructed in this framework in the literature, e.g., [Pas03; BS05; LPV12; Gar+12; Dac+13a; Gar+12; GKP18][4]. Some of these constructions

[3] We note that [GKP18] later constructed a self-composable black-box constant-round general MPC protocol in the SPS framework based on weaker assumptions than our construction. Some of the techniques they used build on techniques presented in this chapter. It is currently unclear, however, if their construction also satisfies our stronger security notion.

[4] Note that this list of references only contains works that constructed general MPC protocols directly in the SPS security framework. See the following paragraphs for constructions proven secure in frameworks with stronger security notions.

are both constant-round and based on standard polynomial-time assumptions: [LPV12] (based on constant-round semi-honest OT), [Gar+12; GKP18] (based on constant-round semi-honest OT plus collision-resistant hash functions) and [GKP17] (based on collision-resistant hash functions plus "weakly certifiable" trapdoor permutations plus lossy encryption schemes plus quasi-polynomially-hard injective one-way functions). The construction in [GKP18] is additionally fully black-box.

The Angel-based security framework, first proposed by [PS04], provides a security notion that implies SPS security and is closed under protocol composition. Several general MPC protocol in the plain model have been constructed in this framework [PS04; MMY06].[5] However, all of these constructions are based on non-standard or super-polynomial-time assumptions.

The UC with super-polynomial helpers framework also provides a security notion that implies SPS security and is closed under protocol composition. This framework was put forward by [CLP10] which also provided a construction in the plain model in this framework that only relies on standard polynomial-time assumptions, namely enhanced trapdoor permutations.[6] Since then, many more general MPC protocols in the plain model have been constructed in this framework [LP12; KMO14; Kiy14; Goy+15; HV16]. The most round-efficient protocols so far that are based on standard polynomial-time assumptions were provided by [Goy+15] and [Kiy14]. [Kiy14] gave a fully black-box construction based on constant-round semi-honest OT that requires $\widetilde{O}(\log^2 n)$ rounds. [Goy+15] gave a non-black-box construction based on enhanced trapdoor permutations requiring $\widetilde{O}(\log n)$ rounds. Some helpers in the literature, e. g., [CLP10; KMO14; Kiy14; Goy+15] come with a feature called "robustness" which guarantees that any attack mounted on a constant-round protocol using this helper can be carried out by a PPT adversary with no access to a helper. Protocols proven secure for robust helpers can be "plugged" into constant-round UC-secure protocols, resulting in protocols secure in the UC with super-polynomial helpers framework. Moreover, [CLP13] constructed a protocol that is secure in the UC with super-polynomial helpers framework and additionally preserves certain security properties of other protocols running in the system. They call such protocols "environmentally friendly".

We note that other security notions in the concurrent setting have been proposed which are not based on the idea of providing simulators with super-polynomial resources. The *multiple ideal query model* [GJO10; GJ13; GGJ13; CGJ15] considers simulators that are allowed to make more than one output query per session to the ideal functionality. All known constructions in this framework require a super-constant number of rounds. Another notion is *input indistinguishability* [MPR06; Gar+12] which guarantees that an adversary cannot decide which inputs have been used by the honest protocol parties. We note that this security notion is not closed under protocol composition and incomparable to ours.

[5][BS05] remarked that their construction can be shown to be secure in the Angel-based security framework but provided no proof of this claim.

[6]We note that in some works, e.g. [Kiy14; KMO14; GKP18], UC with super-polynomial helpers is also referred to as Angel-based security. In this work, however, we use the original names of these two frameworks as proposed by [CLP10] and [PS04], respectively.

4.2 Definitions of the Previous Frameworks

In this section, we briefly recall the definitions of SPS and Angel-based security and UC security with super-polynomial helpers.

4.2.1 SPS Security Framework

In the SPS security framework, a simulator's run-time is not required to be bounded by a polynomial. The adversary and environment are still polynomial-time, however. In the following, we recall the definition of emulation in the SPS security framework (first proposed by [Pas03]).

Definition 4.1 (Security with Super-polynomial Simulators (SPS))**.** *Let π and ϕ be protocols. π is said to emulate ϕ in the* SPS security framework*, denoted by $\pi \geq_{SPS} \phi$, if if for every PPT adversary $\mathcal{A}$, there exists an (not necessarily PPT) adversary $\mathcal{S}$ such that for every PPT environment $\mathcal{Z}$ it holds that*

$$\mathrm{Exec}_{\mathsf{UC}}\big(\pi, \mathcal{A}, \mathcal{Z}\big) \overset{c}{\equiv} \mathrm{Exec}_{\mathsf{UC}}\big(\phi, \mathcal{S}, \mathcal{Z}\big)$$

Obviously, UC security implies SPS security. Another straightforward consequence of the above definition is that transitivity does no longer hold for emulation in the SPS security framework. Furthermore, SPS security is not closed under protocol composition. Hence, the advantages provided by the UC composition theorem, i.e. concurrent security and modular analysis (cf. Section 2.4.5), do not carry over to the SPS framework. Still, using arguments presented in [Can01][7], a composition theorem can be shown in the SPS security framework for the following very restricted case: Let ρ^{ϕ} be a protocol that makes exactly *one* subroutine call to a protocol ϕ. Assume $\pi \geq_{\mathrm{SPS}} \phi$. Then it holds that $\rho^{\pi} \geq_{\mathrm{SPS}} \rho^{\phi}$. Alternatively, one can prove a version of the composition theorem for the case that one instance of ϕ, out of potentially many instances called by ρ, is replaced with an instance of π.

4.2.2 Angel-based Security Framework and UC with super-polynomial Helpers Framework

In the Angel-based security framework, the adverary, environment and simulator are required to run in polynomial time but all have acccess to a super-polynomial oracle Γ called *Imaginary Angel*. The Imaginary Angel Γ takes a query q and then outputs an answer $\Gamma(q, \mathcal{C})$ that may depend on the set $\mathcal{C}$ of corrupted protocol parties.

In the following, we recall the definition of emulation in the Angel-based security framework as proposed by [PS04].

Definition 4.2 (Angel-based Security)**.** *Let π and ϕ be protocols, Γ an Imaginary Angel. π is said to emulate ϕ in the* Γ-Angel-based framework*, denoted by $\pi \geq_{\Gamma\text{-}Angel} \phi$, if for every PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{S}$ such that for every PPT environment $\mathcal{Z}$ it holds that*

$$\mathrm{Exec}_{\mathsf{UC}}(\pi, \mathcal{A}^{\Gamma}, \mathcal{Z}^{\Gamma}) \overset{c}{\equiv} \mathrm{Exec}_{\mathsf{UC}}(\phi, \mathcal{S}^{\Gamma}, \mathcal{Z}^{\Gamma})$$

[7]More specifically, the proof of the "single-instance UC composition theorem" in the alternative proof of the UC composition theorem. Note that the general proof of the UC composition theorem in [Can01] does not carry over to SPS security because a PPT environment cannot internally run the simulator since the latter may run in super-polynomial time.

As with UC security, Angel-based security is transitive and closed under protocol composition (cf. [PS04]). Furthermore, it follows directly from the definitions that Angel-based security implies SPS security.

The UC with super-polynomial helpers framework, first proposed by [CLP10], is based on the *Extended Universal Composability Framework* (EUC), which is an extension of the UC framework where all ITIs in a system of protocols may have access to some global entity such as a globally available PKI or CRS. In the UC with super-polynomial helpers framework, the global entity is called *helper (functionality)* $\mathcal{H}$ and may run in super-polynomial time. $\mathcal{H}$ interacts only with the environment as well as the corrupted parties. Furthermore, $\mathcal{H}$ is informed by the environment as soon as a party is corrupted (cf. [CLP10] for a detailed description of this framework). As with Angel-based security, the security notion implied by the UC with super-polynomial helpers framework is transitive and closed under protocol composition. Furthermore, this security notion also implies SPS security (cf. [CLP10]).

4.3 Shielded Oracles

4.3.1 Definition of the Framework

In this section, we present a new framework that is based on the idea of granting simulators only restricted access to super-polynomial resources. Our framework builds on the UC framework (cf. Section 2.4).

First, we set a few conventions. While the UC framework leaves open how session identifiers (SIDs) and corruptions are organized, we follow the convention that both must be consistent with the hierarchical order of the protocols: The SID of a sub-protocol must be an extension of the SID of the calling protocol (see below). Furthermore, in order to corrupt a sub-party, an adversary must corrupt all parties that are above that sub-party in the protocol hierarchy (i.e. if A is calling party of B then an adversary must first corrupt A before he can corrupt B).

SID Convention. We use the following convention for session identifiers: a session ID $\mathtt{sid}$ is of the form $\mathtt{str}_1||\mathtt{str}_2||\ldots||\mathtt{str}_m$, where $\mathtt{str}_i \in \{0,1\}^*$ and $|| \notin \{0,1\}^*$ is a special delimiter symbol. We call a session ID $\mathtt{sid}'$ an *extension* of a session ID $\mathtt{sid}$ if there exists a session ID $\overline{\mathtt{sid}} \in \{0,1\}^* \cup \{||\}$ such that $\mathtt{sid}' = \mathtt{sid}||\overline{\mathtt{sid}}$, and a *one-step extension* if there exists a bitstring $\mathtt{str} \in \{0,1\}^*$ such that $\mathtt{sid}' = \mathtt{sid}||\mathtt{str}$. We stipulate that the session ID of a sub-protocol must be a one-step extension of its calling protocol.

We relax the UC security notion by introducing a super-polynomial time machine that may aid the simulator. This machine is modeled as a *stateful* oracle $\mathcal{O}$ that is "glued" to an the ideal functionality $\mathcal{F}$. $\mathcal{O}$ may freely interact with the simulator and $\mathcal{F}$. However, the simulator does not "see" the communication between between $\mathcal{O}$ and $\mathcal{F}$. Since the output of the oracle is partially hidden from the simulator, we call $\mathcal{O}$ a *shielded oracle*.

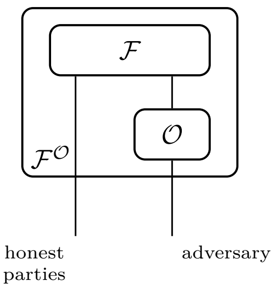


Figure 4.1: $\mathcal{O}$-adjoined functionality $\mathcal{F}^{\mathcal{O}}$ internally runs $\mathcal{F}$ and $\mathcal{O}$ and enforces the correct routing of messages (cp. Definition 4.4)

Definition 4.3 (Shielded Oracles)**.** *A* shielded oracle *is a* stateful *oracle $\mathcal{O}$ that can be implemented in super-polynomial time. By convention, the outputs of a shielded oracle $\mathcal{O}$ are of the form* (`output-to-fnct`, y) *or* (`output-to-adv`, y)*.*

The simulator is allowed to communicate with the ideal functionality *only* via the shielded oracle. This way, the shielded oracle serves as an interface that carries out specific tasks the simulator could not do otherwise. The communication between the shielded oracle and the ideal functionality is hidden away from the simulator. The actions of the shielded oracle may depend on the session identifier (*sid*) of the protocol session as well as the party identifiers of the corrupted parties.

Definition 4.4 ($\mathcal{O}$-adjoined Functionalities)**.** *Given a functionality $\mathcal{F}$ and a shielded oracle $\mathcal{O}$, define the interaction of the $\mathcal{O}$-*adjoined functionality *$\mathcal{F}^{\mathcal{O}}$ in an ideal protocol execution with session identifier sid as follows: (See Fig. 4.1 for a graphical depiction)*

- *$\mathcal{F}^{\mathcal{O}}$ internally runs an instance of $\mathcal{F}$ with session identifier sid*
- *When receiving the first message x from the adversary, $\mathcal{F}^{\mathcal{O}}$ internally invokes $\mathcal{O}$ with input (sid, x).*
  All subsequent messages from the adversary are passed to $\mathcal{O}$.
- *Messages between the honest parties and $\mathcal{F}$ are forwarded.*
- *Corruption messages are forwarded to $\mathcal{F}$ and $\mathcal{O}$.*
- *When $\mathcal{F}$ sends a message y to the adversary, $\mathcal{F}^{\mathcal{O}}$ passes y to $\mathcal{O}$.*
- *The external write operations of $\mathcal{O}$ are treated as follows:*
  - *If $\mathcal{O}$ sends* (`output-to-fnct`, y)*, $\mathcal{F}^{\mathcal{O}}$ sends y to $\mathcal{F}$.*
  - *If $\mathcal{O}$ sends* (`output-to-adv`, y)*, $\mathcal{F}^{\mathcal{O}}$ sends y to the adversary.*

Let IDEAL($\mathcal{F}^{\mathcal{O}}$) be the ideal protocol with functionality $\mathcal{F}^{\mathcal{O}}$ (cf. Section 2.4.1).

In order to obtain a composable security notion, we introduce the notion of *augmented environments.* $\mathcal{F}^{\mathcal{O}}$-augmented environments are UC environments that may invoke, apart form the challenge protocol, polynomially many instances of IDEAL($\mathcal{F}^{\mathcal{O}}$) for a given functionality $\mathcal{F}^{\mathcal{O}}$. The only restriction is that the session identifiers of these instances as well as the session identifier of the challenge protocol are not extensions of one another. $\mathcal{F}^{\mathcal{O}}$-augmented environments may send inputs to and receive outputs from any invoked instance of IDEAL($\mathcal{F}^{\mathcal{O}}$). In addition, $\mathcal{F}^{\mathcal{O}}$-augmented environments can play the role of any adversary via the adversary's interface to the functionality $\mathcal{F}^{\mathcal{O}}$. In particular, $\mathcal{F}^{\mathcal{O}}$-augmented environments may corrupt parties in instances of IDEAL($\mathcal{F}^{\mathcal{O}}$) by sending the corresponding `corrupt` message to the functionality $\mathcal{F}^{\mathcal{O}}$.

In what follows we give the definition of the execution experiment with an $\mathcal{F}^{\mathcal{O}}$-augmented environment.

Definition 4.5 (The $\mathcal{F}^{\mathcal{O}}$-Execution Experiment)**.** *An execution of a protocol σ with adversary $\mathcal{A}$ and an $\mathcal{F}^{\mathcal{O}}$-*augmented environment *$\mathcal{Z}$ on input $a \in \{0,1\}^*$ and with security parameter $n \in \mathbb{N}$ is a run of a system of interactive Turing machines with the following restrictions:*
(See Fig. 4.2 for a graphical depiction)

- *First, $\mathcal{Z}$ is activated on input $a \in \{0,1\}^*$.*
- *The first ITI to be invoked by $\mathcal{Z}$ is the adversary $\mathcal{A}$.*
- *$\mathcal{Z}$ may invoke a single instance of a* challenge protocol, *which is set to be σ by the experiment. The session identifier of σ is determined by $\mathcal{Z}$ upon invocation.*
- *$\mathcal{Z}$ may provide inputs to the adversary or to the main parties of σ.*
- *$\mathcal{Z}$ may provide inputs to and receive subroutine outputs from the main parties of instances of* IDEAL($\mathcal{F}^{\mathcal{O}}$) *as long as the session identifiers of these instances as well as the session identifier of the instance of σ are* not extensions of one another*.*
- *The adversary $\mathcal{A}$ may send messages to the parties of σ as well as give subroutine outputs to $\mathcal{Z}$.*
- *Each party of σ may send messages to $\mathcal{A}$, provide inputs to its subroutines and give subroutine outputs to the parties of which it is a subroutine. Main parties may give subroutine outputs to $\mathcal{Z}$.*
- *The ITIs take turns during the execution experiment, i.e., whenever an ITI writes an allowed external write instruction, then the targeted ITI is activated and the sending ITI is suspended. If an ITI suspends its computation without writing an external write instruction or if it writes a disallowed external write instruction, then the environment $\mathcal{Z}$ is activated. (Note that this has the effect that at any point in time throughout the execution experiment only a single ITI is active.)*
- *At the end of the execution experiment, $\mathcal{Z}$ outputs a single bit.*

Denote by $\mathrm{Exec}\big(\sigma, \mathcal{A}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]\big)(n,a)$ *the random variable defined as the output of the $\mathcal{F}^{\mathcal{O}}$-augmented environment $\mathcal{Z}$ on input $a \in \{0,1\}^*$ and with security parameter $n \in \mathbb{N}$ after interacting with σ and $\mathcal{A}$ according to the above definition.*

Define $\mathrm{Exec}\big(\sigma, \mathcal{A}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]\big) = \big\{\mathrm{Exec}\big(\sigma, \mathcal{A}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]\big)(n,a)\big\}_{n\in\mathbb{N}, a\in\{0,1\}^*}$

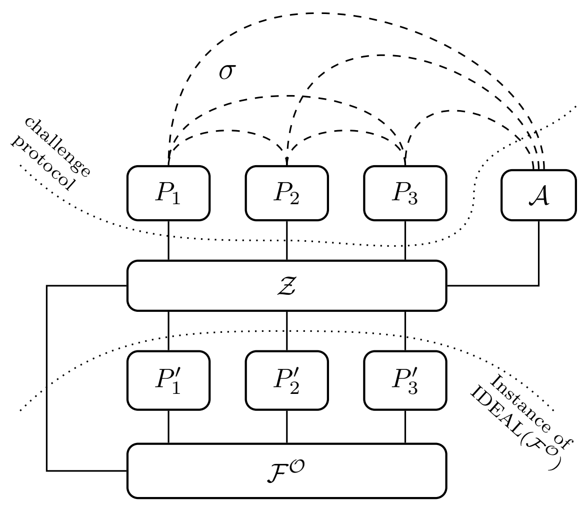


Figure 4.2: Execution of the real experiment with challenge protocol and one additionally invoked instance of IDEAL($\mathcal{F}^{\mathcal{O}}$) (cp. Definition 4.5)

We will now define security in our framework in total analogy to the UC framework:

Definition 4.6 ($\mathcal{F}^{\mathcal{O}}$-Emulation)**.** *Let π and ϕ be protocols. π is said to emulate ϕ in the presence of $\mathcal{F}^{\mathcal{O}}$-augmented environments, denoted by $\pi \geq_{\mathcal{F}^{\mathcal{O}}} \phi$, if for any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{S}$ such that for every $\mathcal{F}^{\mathcal{O}}$-augmented PPT environment $\mathcal{Z}$ it holds that*

$$\operatorname{Exec}\big(\pi, \mathcal{A}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]\big) \overset{c}{\equiv} \operatorname{Exec}\big(\phi, \mathcal{S}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]\big)$$

As in the UC framework, we will sometimes abuse notation and write $\pi \geq_{\mathcal{F}^{\mathcal{O}}} \mathcal{F}^{\mathcal{O}}$ if π emulates the ideal protocol with $\mathcal{O}$-adjoined functionality $\mathcal{F}^{\mathcal{O}}$ in the presence of $\mathcal{F}^{\mathcal{O}}$-augmented environments.

Throughout this chapter, we only consider *static* corruptions. (Note that in the case of static corruptions, an $\mathcal{F}^{\mathcal{O}}$-augmented environment determines the set of corrupted parties of each instance of IDEAL($\mathcal{F}^{\mathcal{O}}$) when invoking that instance.)

4.3.2 Basic Properties and Justification

In this section, we show that our security notion is transitive and that the dummy adversary is complete within this notion. As a justification for our new security notion, we show that it implies SPS security.

Definition 4.7 ($\mathcal{F}^{\mathcal{O}}$-Emulation with Respect to the Dummy Adversary)**.** *The* dummy adversary *$\mathcal{D}$ is an adversary that when receiving a message (sid, pid, m) from the environment, sends m to the party with party identifier pid and session* *identifier sid, and that, when receiving m from the party with party identifier pid and session identifier sid, sends $(\mathit{sid}, \mathit{pid}, m)$ to the environment.*

Let π and ϕ be protocols. π is said to emulate ϕ in the presence of $\mathcal{F}^{\mathcal{O}}$-augmented environments with respect to the dummy adversary*, if*

$$\exists\, \mathcal{S}_{\mathcal{D}}\ \forall\, \mathcal{Z} : \mathrm{Exec}\big(\pi, \mathcal{D}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]\big) \overset{\mathrm{c}}{\equiv} \mathrm{Exec}\big(\phi, \mathcal{S}_{\mathcal{D}}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]\big) \ .$$

Proposition 4.8 (Completeness of the Dummy Adversary)**.** *Let π and ϕ be protocols. Then, π emulates ϕ in the presence of $\mathcal{F}^{\mathcal{O}}$-augmented environments if and only if π emulates ϕ in the presence of $\mathcal{F}^{\mathcal{O}}$-augmented environments with respect to the dummy adversary.*

The proof is almost exactly the same as in [Can01]. We give the proof here for the sake of completeness.

Proof. Clearly, if π emulates ϕ in the presence of $\mathcal{F}^{\mathcal{O}}$-augmented environments then π also emulates ϕ in the presence of $\mathcal{F}^{\mathcal{O}}$-augmented environments with respect to the dummy adversary.

We now show the converse. Let π emulate ϕ in the presence of $\mathcal{F}^{\mathcal{O}}$-augmented environments with respect to the dummy adversary. Let $\mathcal{A}$ be an adversary interacting the protocol π and an $\mathcal{F}^{\mathcal{O}}$-augmented environment $\mathcal{Z}$. Define the simulator $\mathcal{S}_{\mathcal{A}}$ as follows: $\mathcal{S}_{\mathcal{A}}$ internally runs simulated copies of $\mathcal{A}$ and the dummy adversary simulator $\mathcal{S}_{\mathcal{D}}$. $\mathcal{S}_{\mathcal{A}}$ forwards the communication between $\mathcal{A}$ and $\mathcal{Z}$ and between $\mathcal{S}_{\mathcal{D}}$ and ϕ. When $\mathcal{A}$ delivers a message m to a party with party identifier *pid* and session identifier *sid*, $\mathcal{S}_{\mathcal{A}}$ sends $(\mathit{sid}, \mathit{pid}, m)$ to $\mathcal{S}_{\mathcal{D}}$. When $\mathcal{S}_{\mathcal{D}}$ outputs $(\mathit{sid}, \mathit{pid}, m)$, $\mathcal{S}_{\mathcal{A}}$ copies m to the incoming communication tape of $\mathcal{A}$ as a message coming from the party with party identifier *pid* and session identifier *sid*.

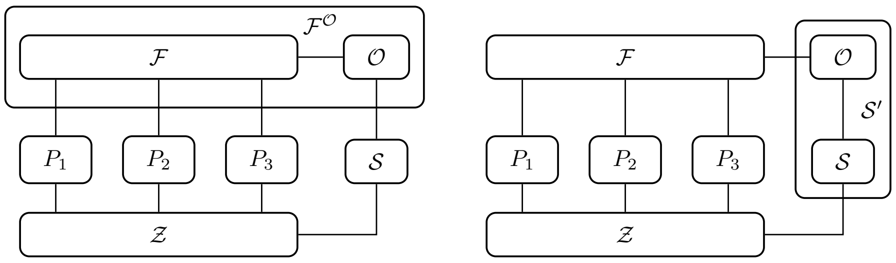
Next define the environment $\mathcal{Z}_{\mathcal{A}}$ as follows: $\mathcal{Z}_{\mathcal{A}}$ internally runs simulated copies of $\mathcal{Z}$ and $\mathcal{A}$. $\mathcal{Z}_{\mathcal{A}}$ forwards the communication between $\mathcal{A}$ and $\mathcal{Z}$ and between $\mathcal{Z}$ and the protocol parties. When $\mathcal{A}$ delivers a message m to the party with party identifier *pid* and session identifier *sid*, $\mathcal{Z}_{\mathcal{A}}$ sends $(\mathit{sid}, \mathit{pid}, m)$ to the dummy adversary. Likewise, when $\mathcal{Z}_{\mathcal{A}}$ receives a message $(\mathit{sid}, \mathit{pid}, m)$ from the dummy adversary, $\mathcal{Z}_{\mathcal{A}}$ copies m to the incoming communication tape of $\mathcal{A}$ as a message coming from the party with party identifier *pid* and session identifier *sid*. Furthermore, $\mathcal{Z}_{\mathcal{A}}$ invokes the same instances of IDEAL($\mathcal{F}^{\mathcal{O}}$) that the environment $\mathcal{Z}$ invokes. Finally, $\mathcal{Z}_{\mathcal{A}}$ outputs whatever $\mathcal{Z}$ outputs. We have that

$$\begin{aligned}\mathrm{Exec}\big(\pi, \mathcal{A}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]\big) &= \mathrm{Exec}\big(\pi, \mathcal{D}, \mathcal{Z}_{\mathcal{A}}[\mathcal{F}^{\mathcal{O}}]\big)\\ &\overset{\mathrm{c}}{\equiv} \mathrm{Exec}\big(\phi, \mathcal{S}_{\mathcal{D}}, \mathcal{Z}_{\mathcal{A}}[\mathcal{F}^{\mathcal{O}}]\big)\\ &= \mathrm{Exec}\big(\phi, \mathcal{S}_{\mathcal{A}}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]\big) \ ,\end{aligned}$$

where the second step uses the premise. The statement follows. □

Next, we show that our notion is transitive.

Proposition 4.9 (Transitivity)**.** *Let π_1, π_2, π_3 be protocols. If $\pi_1 \geq_{\mathcal{F}^{\mathcal{O}}} \pi_2$ and $\pi_2 \geq_{\mathcal{F}^{\mathcal{O}}} \pi_3$, then it holds that $\pi_1 \geq_{\mathcal{F}^{\mathcal{O}}} \pi_3$.*



(a) The super-polynomial power of $\mathcal{O}$ is encapsulated inside of $\mathcal{F}^{\mathcal{O}}$

(b) The super-polynomial simulator $\mathcal{S}'$ internally runs the code of $\mathcal{O}$ itself

Figure 4.3: Shielded Oracles imply SPS security (cp. Proposition 4.10)

Proof. Let $\mathcal{A}$ be an adversary interacting with π_1. Since $\pi_1 \geq_{\mathcal{F}^{\mathcal{O}}} \pi_2$, there exists a simulator $\mathcal{S}_1$ such that $\text{Exec}(\pi_1, \mathcal{A}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]) \overset{c}{\equiv} \text{Exec}(\pi_2, \mathcal{S}_1, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}])$, and since $\pi_2 \geq_{\mathcal{F}^{\mathcal{O}}} \pi_3$, there exists a simulator $\mathcal{S}_2$ such that $\text{Exec}(\pi_2, \mathcal{S}_1, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]) \overset{c}{\equiv} \text{Exec}(\pi_3, \mathcal{S}_2, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}])$ for all $\mathcal{F}^{\mathcal{O}}$-augmented environments $\mathcal{Z}$.

Thus, $\text{Exec}(\pi_1, \mathcal{A}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]) \overset{c}{\equiv} \text{Exec}(\pi_3, \mathcal{S}_2, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}])$[8]

□

As a justification for our new notion, one can show that security with respect to $\mathcal{F}^{\mathcal{O}}$-emulation implies security with respect to SPS emulation. The proof is straightforward: View the oracle as part of the simulator. This simulator runs in super-polynomial time, hence can be simulated by an SPS-simulator (cf. Fig. 4.3 for a graphical depiction).

Proposition 4.10 ($\mathcal{F}^{\mathcal{O}}$-Emulation implies SPS Emulation)**.** *Let $\mathcal{O}$ be a shielded oracle. Assume $\pi \geq_{\mathcal{F}^{\mathcal{O}}} \mathcal{F}^{\mathcal{O}}$. Then it holds that $\pi \geq_{SPS} \mathcal{F}$.*

4.3.3 Universal Composition

In this section, we prove that the security notion provided by our framework is closed under protocol composition. More specifically, we generalize the UC composition theorem to also include $\mathcal{F}^{\mathcal{O}}$-hybrid protocols.

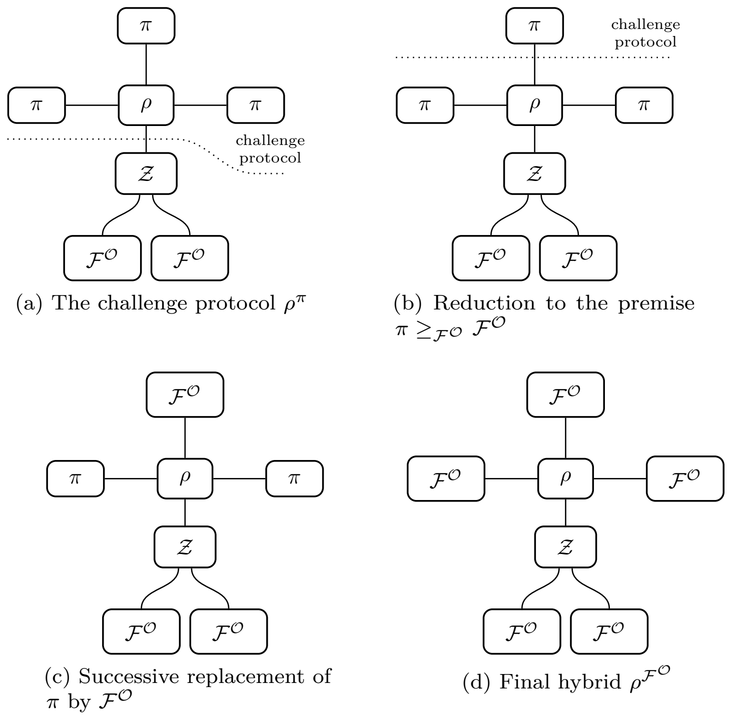
Theorem 4.11 (Composition Theorem)**.** *Let $\mathcal{O}$ be a shielded oracle, $\mathcal{F}$ and $\mathcal{G}$ functionalities.*

1. (Polynomial hybrid protocols) *Let π, $\rho^{\mathcal{G}}$ be protocols. Assume $\pi \geq_{\mathcal{F}^{\mathcal{O}}} \mathcal{G}$. Then it holds that $\rho^{\pi} \geq_{\mathcal{F}^{\mathcal{O}}} \rho^{\mathcal{G}}$.*

2. ($\mathcal{F}^{\mathcal{O}}$-hybrid protocols) *Let π be a protocol, $\rho^{\mathcal{F}^{\mathcal{O}}}$ a protocol in the $\mathcal{F}^{\mathcal{O}}$-hybrid model. Assume $\pi \geq_{\mathcal{F}^{\mathcal{O}}} \mathcal{F}^{\mathcal{O}}$. Then it holds that $\rho^{\pi} \geq_{\mathcal{F}^{\mathcal{O}}} \rho^{\mathcal{F}^{\mathcal{O}}}$*

Proof Idea. The first statement follows from the same argument as in the proof for the UC composition theorem, see [Can01].

[8]Note that our hierarchical SID convention implies that the class of $\mathcal{F}^{\mathcal{O}}$-augmented environments that try to distinguish between π_1 and π_2 is the same as the class of $\mathcal{F}^{\mathcal{O}}$-augmented environments that try to distinguish between π_2 and π_3.



(a) The challenge protocol ρ^{π}

(b) Reduction to the premise $\pi \geq_{\mathcal{F}^{\mathcal{O}}} \mathcal{F}^{\mathcal{O}}$

(c) Successive replacement of π by $\mathcal{F}^{\mathcal{O}}$

(d) Final hybrid $\rho^{\mathcal{F}^{\mathcal{O}}}$

Figure 4.4: Proof of the composition theorem (Theorem 4.11)

Second statement: This statement also follows from very similar arguments as in the proof for the UC composition theorem. In the following, we very briefly sketch some of these arguments (note that in the following, we use arguments that are similiar to the simpler "alternative proof" of the UC theorem, cf. [Can01]).

Let $\mathcal{Z}$ be an $\mathcal{F}^{\mathcal{O}}$-augmented environment interacting with ρ^{π} and the dummy adversary $\mathcal{D}$. Using a standard hybrid argument, iteratively replace all instances of π called by ρ with instances of IDEAL($\mathcal{F}^{\mathcal{O}}$). Indistinguishability follows by reduction to the premise $\pi \geq_{\mathcal{F}^{\mathcal{O}}} \mathcal{F}^{\mathcal{O}}$. In the reduction, the remaining instances of π and ρ are treated as part of an $\mathcal{F}^{\mathcal{O}}$-augmented environment which internally runs $\mathcal{Z}$ and invokes the remaining instances of IDEAL($\mathcal{F}^{\mathcal{O}}$) (see Fig. 4.4 for a graphical depiction). The claim then follows from the completeness of the dummy adversary (Proposition 4.8).

□

The universal composition theorem in the UC framework has two important implications (cf. Section 2.4.5): concurrent security and modular analysis. The former means that the security properties of the analyzed protocol remain valid even when multiple instances of the protocol are concurrently executed in an unknown environment. The latter implies that one can deduce the security of a composite protocol from its components.

Theorem 4.11 directly implies concurrent security (with super-polynomial time simulators). However, modular analysis is not directly implied by Theorem 4.11. This is because the oracle $\mathcal{O}$ *may* contain all "complexity" of the protocol π, i. e., proving the security of $\rho^{\mathcal{F}^{\mathcal{O}}}$ may be as complex as proving the security of ρ^{π}.

Still, one can use Theorem 4.11 to achieve modular analysis by constructing secure protocols with "strong modular composition properties". A protocol π with such properties can be "plugged" into a large class of UC-secure protocols, such that the composite protocol is secure in our framework. This allows analyzing the security of a large class of protocols $\rho^{\mathcal{F}}$ in the UC framework and achieve security in our framework when replacing $\mathcal{F}$ with π. As a proof of concept, we will show, using Theorem 4.11, that a large class of protocols in the $\mathcal{F}_{\mathrm{com}}$-hybrid model can be composed with a commitment protocol presented in this chapter (cf. Theorem 4.34).

Remark 4.12. *Note that Proposition 4.10 and Theorem 4.11 imply that the security notion of our framework is strictly stronger than SPS security since the latter is not closed under protocol composition. For a concrete separating example, we refer to Remark 4.27.*

The following is a useful extension of Theorem 4.11 for multiple oracles.

Corollary 4.13 (Composition Theorem for Multiple Oracles)**.** *Let $\mathcal{O}$, $\mathcal{O}'$ be shielded oracles. Assume that $\pi \geq_{\mathcal{F}^{\mathcal{O}}} \mathcal{F}^{\mathcal{O}}$ and $\rho^{\mathcal{F}^{\mathcal{O}}} \geq_{\mathcal{F}^{\mathcal{O}},\mathcal{G}^{\mathcal{O}'}} \mathcal{G}^{\mathcal{O}'}$. Then there exists a shielded oracle $\mathcal{O}''$ such that $\rho^{\pi} \geq_{\mathcal{G}^{\mathcal{O}''}} \mathcal{G}^{\mathcal{O}''}$.*

Proof. Since $\rho^{\mathcal{F}^{\mathcal{O}}}$ emulates $\mathcal{G}^{\mathcal{O}'}$, there exists a simulator $\mathcal{S}_{\mathcal{D}}$ for the dummy adversary.

Define $\mathcal{O}''$ as follows: $\mathcal{O}''$ internally simulates $\mathcal{S}_{\mathcal{D}}$ and $\mathcal{O}'$, passes each message $\mathcal{S}_{\mathcal{D}}$ sends to $\mathcal{G}$ to $\mathcal{O}'$, sends each `output-to-fnct` output from $\mathcal{O}'$ to $\mathcal{G}$ and each `output-to-adv` output to $\mathcal{S}_{\mathcal{D}}$, and forwards the communication between $\mathcal{S}_{\mathcal{D}}$ and the environment. By construction, it holds that

$$\begin{aligned}\mathrm{Exec}\big(\rho^{\mathcal{F}^{\mathcal{O}}},\mathcal{D},\mathcal{Z}[\mathcal{F}^{\mathcal{O}},\mathcal{G}^{\mathcal{O}'}]\big) &\overset{\mathrm{c}}{\equiv} \mathrm{Exec}\big(\mathcal{G}^{\mathcal{O}'},\mathcal{S}_{\mathcal{D}},\mathcal{Z}[\mathcal{F}^{\mathcal{O}},\mathcal{G}^{\mathcal{O}'}]\big)\\ &\equiv \mathrm{Exec}\big(\mathcal{G}^{\mathcal{O}''},\mathcal{D},\mathcal{Z}[\mathcal{F}^{\mathcal{O}},\mathcal{G}^{\mathcal{O}'}]\big)\end{aligned}$$

Since $\mathcal{S}_{\mathcal{D}}$ runs in polynomial time, $(\mathcal{F}^{\mathcal{O}},\mathcal{G}^{\mathcal{O}'})$-augmented environments can simulate $(\mathcal{F}^{\mathcal{O}},\mathcal{G}^{\mathcal{O}''})$-augmented environments. Therefore, it follows from Proposition 4.8 that $\rho^{\mathcal{F}^{\mathcal{O}}} \geq_{\mathcal{F}^{\mathcal{O}},\mathcal{G}^{\mathcal{O}''}} \mathcal{G}^{\mathcal{O}''}$ and $\mathcal{G}^{\mathcal{O}''} \geq_{\mathcal{F}^{\mathcal{O}},\mathcal{G}^{\mathcal{O}''}} \rho^{\mathcal{F}^{\mathcal{O}}}$. By the composition theorem we have that $\rho^{\pi} \geq_{\mathcal{F}^{\mathcal{O}}} \rho^{\mathcal{F}^{\mathcal{O}}}$. Hence

$$\rho^{\pi} \underset{\mathcal{F}^{\mathcal{O}}}{\geq} \rho^{\mathcal{F}^{\mathcal{O}}} \underset{\mathcal{G}^{\mathcal{O}''}}{\geq} \mathcal{G}^{\mathcal{O}''}$$

Since it holds that $\mathcal{G}^{\mathcal{O}''} \geq_{\mathcal{F}^{\mathcal{O}},\mathcal{G}^{\mathcal{O}''}} \rho^{\mathcal{F}^{\mathcal{O}}}$, one can iteratively replace all instances of IDEAL($\mathcal{G}^{\mathcal{O}''}$) invoked by a $\mathcal{G}^{\mathcal{O}''}$-augmented environment with instances of IDEAL($\rho^{\mathcal{F}^{\mathcal{O}}}$), obtaining an $\mathcal{F}^{\mathcal{O}}$-augmented environment. Therefore, it holds that $\rho^{\pi} \geq_{\mathcal{G}^{\mathcal{O}''}} \rho^{\mathcal{F}^{\mathcal{O}}}$. The statement follows from the transitivity of $\mathcal{G}^{\mathcal{O}''}$-emulation. □

4.3.4 Polynomial Simulatability

We show a unique feature of our framework: For appropriate oracles to be defined below, augmented environments do not "hurt" UC-secure protocols. This means that a protocol that was proven secure in the UC framework is also secure in our framework. As a consequence, our security notion is fully compatible with UC security.

Definition 4.14 (Polynomial Simulatability)**.** *Let $\mathcal{O}$ be a shielded oracle, $\mathcal{F}$ a functionality. Say that $\mathcal{F}^{\mathcal{O}}$ is* polynomially simulatable *if there exists a PPT functionality $\mathcal{M}$ such that*

$$\mathcal{F}^{\mathcal{O}} \underset{\mathcal{F}^{\mathcal{O}}}{\geq} \mathcal{M}$$

If a functionality $\mathcal{F}^{\mathcal{O}}$ is polynomially simulatable then the super-polynomial power of the oracle $\mathcal{O}$ is totally "shielded away" from the environment. Note that in Definition 4.14, indistinguishability must hold for *augmented* environments not only for polynomial-time environments.

As a consequence, $\mathcal{F}^{\mathcal{O}}$-augmented environments can be replaced by *polynomial-time* environments if $\mathcal{F}^{\mathcal{O}}$ is polynomially simulatable.

Theorem 4.15 (Reduction to polynomial time Environments)**.** *Let $\mathcal{O}$ be a shielded oracle and $\mathcal{F}$ a functionality such that $\mathcal{F}^{\mathcal{O}}$ is polynomially simulatable. Let π, ϕ be protocols that are PPT or in the $\mathcal{F}^{\mathcal{O}}$-hybrid model. It holds that*

$$\pi \underset{\mathcal{F}^{\mathcal{O}}}{\geq} \phi \iff \pi \underset{\mathsf{poly}}{\geq} \phi$$

where the right-hand side means that π emulates ϕ in the presence of all $\mathcal{F}^{\mathcal{O}}$-augmented environments that never invoke an instance of IDEAL($\mathcal{F}^{\mathcal{O}}$) *(outside of the challenge protcol).*

Proof Idea. Poly-emulation implies $\mathcal{F}^{\mathcal{O}}$-emulation: Since $\mathcal{F}^{\mathcal{O}}$ is polynomially simulatable, there exists a PPT functionality $\mathcal{M}$ such that $\mathcal{F}^{\mathcal{O}} \geq_{\mathcal{F}^{\mathcal{O}}} \mathcal{M}$. Given a $\mathcal{F}^{\mathcal{O}}$-augmented environment $\mathcal{Z}$, replace all instances of IDEAL($\mathcal{F}^{\mathcal{O}}$) which are invoked by $\mathcal{Z}$ outside of the challenge protocol with instances of the ideal protocol with functionality $\mathcal{M}$. Indistinguishability follows by reduction to $\pi \geq_{\mathcal{F}^{\mathcal{O}}} \mathcal{F}^{\mathcal{O}}$. Treat all instances of $\mathcal{M}$ as part of a new environment $\mathcal{Z}'$ which internally runs $\mathcal{Z}$. It holds that $\mathcal{Z}'$ is PPT because $\mathcal{M}$ and $\mathcal{Z}$ are PPT. Then use the premise $\pi \geq_{\mathsf{poly}} \phi$.

The converse is trivial. □

Since $\mathcal{F}^{\mathcal{O}}$-augmented environments that never invoke instances of IDEAL($\mathcal{F}^{\mathcal{O}}$) (outside of the challenge protcol) are identical to UC-environments, the following corollary immediately follows.

Corollary 4.16 (Compatibility with the UC Framework)**.** *Let $\mathcal{O}$ be a shielded oracle and $\mathcal{F}$ a functionality such that $\mathcal{F}^{\mathcal{O}}$ is polynomially simulatable. It holds that*

$$\pi \underset{\mathcal{F}^{\mathcal{O}}}{\geq} \phi \iff \pi \underset{\mathsf{UC}}{\geq} \phi$$

Note that this does not contradict the classical impossibility results of the UC framework (e.g. [CF01]): If $\pi \geq_{\mathcal{F}^{\mathcal{O}}} \mathcal{F}^{\mathcal{O}}$ for a polynomially simulatable $\mathcal{F}^{\mathcal{O}}$, then this only implies that $\pi \geq_{\mathsf{UC}} \mathcal{F}^{\mathcal{O}}$, but it does not follow that $\pi \geq_{\mathsf{UC}} \mathcal{F}$. Although the super-polynomial power of $\mathcal{O}$ is shielded away from the outside, it is indeed necessary.

Replacing augmented environments with polynomial-time environments will be a key property in various proofs later in this chapter. In particular, it will allow us to prove the security of protocols in our framework using the relatively weak primitive *parallel* CCA-secure commitmens as opposed to CCA-secure commitments, which were used in constructions in the Angel-based security and UC with super-polynomial helpers framework.

Remark 4.17. *Note that a shielded oracle $\mathcal{O}$ can still "hurt" the security of a functionality $\mathcal{F}$ even if $\mathcal{F}^{\mathcal{O}}$ is polynomially simulatable. Consider the following example: Let $\mathcal{F}$ be the two-party functionality that receives a group element g from party A and an integer x from party B and outputs (g, g^x) to A and the adversary. Let $\mathcal{O}$ be the oracle that when receiving (g, g^x) computes the discrete logarithm x of g^x to the base g and outputs x to the adversary. Furthermore, let $\mathcal{M}$ be the PPT functionality that on input g and x from A and B outputs (g, g^x) to A and (g, x) to the adversary. It holds that $\mathcal{F}^{\mathcal{O}} \geq_{\mathcal{F}^{\mathcal{O}}} \mathcal{M}$. Hence, $\mathcal{F}^{\mathcal{O}}$ is polynomially simulatable but the adversary learns the secret input of B.*

Next, we show that by suitably tweaking a given oracle $\mathcal{O}$ one can make $\mathcal{F}^{\mathcal{O}}$ polynomially simulatable while preserving the security relation.

Lemma 4.18 (Derived Oracle)**.** *Let $\mathcal{O}$ be a shielded oracle such that $\pi \geq_{\mathcal{F}^{\mathcal{O}}} \mathcal{F}^{\mathcal{O}}$. Then there exists a shielded oracle $\widetilde{\mathcal{O}}$ such that $\pi \geq_{\mathcal{F}^{\widetilde{\mathcal{O}}}} \mathcal{F}^{\widetilde{\mathcal{O}}}$ and* additionally *$\mathcal{F}^{\widetilde{\mathcal{O}}}$ is polynomially simulatable.*

Proof. (See Fig. 4.5 for a graphical depiction of the proof.) Since π emulates $\mathcal{F}^{\mathcal{O}}$, there exists a simulator $\mathcal{S}_{\mathcal{D}}$ for the dummy adversary $\mathcal{D}$. Define the shielded oracle $\widetilde{\mathcal{O}}$ as follows: $\widetilde{\mathcal{O}}$ internally simulates $\mathcal{S}_{\mathcal{D}}$ and $\mathcal{O}$, passes each message $\mathcal{S}_{\mathcal{D}}$ sends to $\mathcal{F}$ to $\mathcal{O}$, sends each `output-to-fnct` output from $\mathcal{O}$ to $\mathcal{F}$ and each `output-to-adv` output from $\mathcal{O}$ to $\mathcal{S}_{\mathcal{D}}$, and forwards the communication between $\mathcal{S}_{\mathcal{D}}$ and the environment.

By construction, for all $\mathcal{F}^{\mathcal{O}}$-augmented environments $\mathcal{Z}$ it holds that

$$\mathrm{Exec}(\pi, \mathcal{D}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]) \overset{c}{\equiv} \mathrm{Exec}(\mathcal{F}^{\mathcal{O}}, \mathcal{S}_{\mathcal{D}}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]) = \mathrm{Exec}(\mathcal{F}^{\widetilde{\mathcal{O}}}, \mathcal{D}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}])$$

It follows from Proposition 4.8 that $\pi \geq_{\mathcal{F}^{\mathcal{O}}} \mathcal{F}^{\widetilde{\mathcal{O}}}$ and $\mathcal{F}^{\widetilde{\mathcal{O}}} \geq_{\mathcal{F}^{\mathcal{O}}} \pi$. Since $\mathcal{S}_{\mathcal{D}}$ runs in polynomial time, $\mathcal{F}^{\mathcal{O}}$-augmented environments can simulate $\mathcal{F}^{\widetilde{\mathcal{O}}}$-augmented environments. Therefore, $\pi \geq_{\mathcal{F}^{\widetilde{\mathcal{O}}}} \mathcal{F}^{\widetilde{\mathcal{O}}}$ and $\mathcal{F}^{\widetilde{\mathcal{O}}} \geq_{\mathcal{F}^{\widetilde{\mathcal{O}}}} \pi$. The theorem follows by defining $\mathcal{M}$ to be the functionality that internally simulates the protocol π.

□

The following corollary shows that UC-secure protocols can be used as sub-protocols in protocols proven secure in our framework, while preserving security.

Corollary 4.19 (Composition with UC-secure Protocols)**.** *Let π, $\rho^{\mathcal{F}}$ be protocols such that $\pi \geq_{\mathsf{UC}} \mathcal{F}$ and $\rho^{\mathcal{F}} \geq_{\mathcal{G}^{\mathcal{O}}} \mathcal{G}^{\mathcal{O}}$. Then there exists a shielded oracle $\mathcal{O}'$ such that*

$$\rho^{\pi} \underset{\mathcal{G}^{\mathcal{O}'}}{\geq} \mathcal{G}^{\mathcal{O}'}$$

Proof. Since $\rho^{\mathcal{F}}$ is PPT there exists a shielded oracle $\mathcal{O}'$ such that $\mathcal{G}^{\mathcal{O}'}$ is polynomially simulatable and $\rho^{\mathcal{F}} \geq_{\mathcal{G}^{\mathcal{O}'}} \mathcal{G}^{\mathcal{O}'}$ by Lemma 4.18. From Corollary 4.16

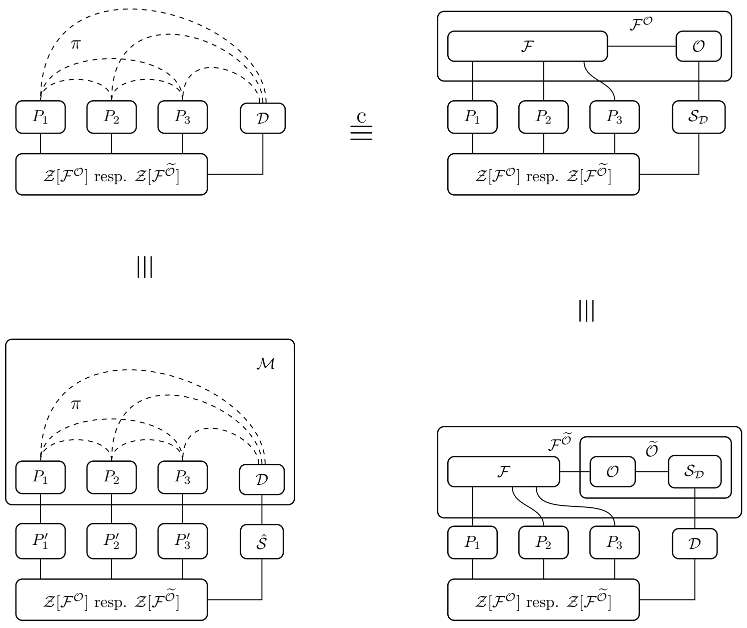

π
P_1
P_2
P_3
$\mathcal{D}$
$\mathcal{Z}[\mathcal{F}^{\mathcal{O}}]$ resp. $\mathcal{Z}[\mathcal{F}^{\widetilde{\mathcal{O}}}]$
$\stackrel{c}{\equiv}$
$\mathcal{F}^{\mathcal{O}}$
$\mathcal{F}$
$\mathcal{O}$
$\mathcal{S}_{\mathcal{D}}$
$\equiv$
$\mathcal{M}$
P'_1
P'_2
P'_3
$\hat{\mathcal{S}}$
$\mathcal{F}^{\widetilde{\mathcal{O}}}$
$\widetilde{\mathcal{O}}$

Figure 4.5: Derived oracle (cp. Lemma 4.18)

it follows that $\pi \geq_{\mathcal{G}^{\mathcal{O}'}} \mathcal{F}$. The statement then follows from the composition theorem and the transitivity of $\mathcal{G}^{\mathcal{O}'}$-emulation. $\square$

The last result demonstrates the compatibility of our framework with the UC framework again. Note that while it is much more desireable to "plug" a protocol proven secure in our framework into a UC-secure protocol—in order to obtain a secure protocol in the *plain model* (this will be addressed in Theorem 4.34 and Corollary 4.38)—doing it the other way around is still a convenient property. For instance, it allows instantiating "auxiliary" functionalities such as authenticated message transmission or secure message transmission, while preserving security.

4.3.5 Relation with Angel-based Security

A natural question that arises is how our security notion compares to Angel-based security. We will prove that for a large class of Imaginary Angels – which to the best of our knowledge includes all Imaginary Angels that can be found in the literature – Angel-based security implies our security notion. However, assuming the existence of one-way functions, the converse does not hold. Thus, our notion is *strictly weaker* than Angel-based security.[9]

In the following, we define the class of Imaginary Angels for which it holds that Angel-based security implies our security notion.[10]

[9]Note that the following results can be generalized to also hold for "UC security with super-polynomial helpers", see [Bro+17] for a proof.

[10] [Bro+17] considered a more general class. However, the definition of this class is quite involved. We have therefore chosen to present a proof for a smaller class that is easier to define.

Definition 4.20 (Normal Imaginary Angel)**.** *An Imaginary Angel Γ is called* normal *if for any query q and any two sets $\mathcal{C}, \mathcal{C}'$ of corrupted parties such that $\mathcal{C} \subseteq \mathcal{C}'$ it holds that $\Gamma(q, \mathcal{C}) = \Gamma(q, \mathcal{C}')$.*

Theorem 4.21 (Relation between Angel-based Security and Shielded Oracles)**.**

1. *Assume $\pi \geq_{\Gamma\text{-}Angel} \mathcal{F}$ for an Imaginary Angel Γ. If Γ is normal, then there exists a shielded oracle $\mathcal{O}$ such that $\pi \geq_{\mathcal{F}^{\mathcal{O}}} \mathcal{F}^{\mathcal{O}}$.*

2. *Assume the existence of one-way functions. Then there exists a protocol ρ (in the $\mathcal{F}_{auth}$-hybrid model), a functionality $\mathcal{G}$ and a shielded oracle $\mathcal{O}$ such that $\rho \geq_{\mathcal{G}^{\mathcal{O}}} \mathcal{G}^{\mathcal{O}}$ but no Imaginary Angel Γ can be found such that $\rho \geq_{\Gamma\text{-}Angel} \mathcal{G}$ holds.*

Proof Sketch. 1. Let $\rho^{\mathcal{F}}$ be the protocol that consists of one instance of the ideal protocol with functionality $\mathcal{F}$ and polynomially many instances of a protocol λ. The protocol λ is the ideal protocol with a functionality that ignores all inputs coming from the parties and outputs a notification message, say $\perp$, to one of the parties when receiving a message from the adversary. Instances of λ are invoked through special invocation inputs provided by the environment. Let ρ^{π} be like $\rho^{\mathcal{F}}$, except that the one instance of the ideal protocol with functionality $\mathcal{F}$ is replaced with an instance of π. Since $\pi \geq_{\Gamma\text{-Angel}} \mathcal{F}$, it follows from the composition theorem of the Angel-based security framework that $\rho^{\pi} \geq_{\Gamma\text{-Angel}} \rho^{\mathcal{F}}$. Hence, there exists a simulator $\mathcal{S}_{\mathcal{D}}$ such that for all environments $\mathcal{Z}^{\Gamma}$ it holds that $\mathrm{Exec}_{\mathsf{UC}}(\rho^{\pi}, \mathcal{D}, \mathcal{Z}^{\Gamma}) \overset{c}{\equiv} \mathrm{Exec}_{\mathsf{UC}}(\rho^{\mathcal{F}}, \mathcal{S}_{\mathcal{D}}^{\Gamma}, \mathcal{Z}^{\Gamma})$.

Define the shielded oracle $\mathcal{O}$ as follows: $\mathcal{O}$ internally runs $\mathcal{S}_{\mathcal{D}}$ and Γ. $\mathcal{O}$ forwards all messages from the ideal functionality to $\mathcal{S}_{\mathcal{D}}$. Messages coming from the environment are relayed by $\mathcal{O}$ to $\mathcal{S}_{\mathcal{D}}^{\Gamma}$ if they are addressed to the instance of π and ignored otherwise. $\mathcal{O}$ answers the oracle queries of $\mathcal{S}_{\mathcal{D}}$ by running Γ. If $\mathcal{S}_{\mathcal{D}}$ sends a message to the functionality or to the environment, $\mathcal{O}$ forwards this message to the respective ITI. Messages sent by $\mathcal{S}_{\mathcal{D}}$ to other ITIs are ignored.

Given an $\mathcal{F}^{\mathcal{O}}$-augmented environment $\mathcal{Z}$, define the Γ-Angel environment $\mathcal{Z}'$ as follows: $\mathcal{Z}'$ internally runs $\mathcal{Z}$. Messages between the parties of π and $\mathcal{Z}$ are relayed. If $\mathcal{Z}$ sends a message to the adversary, $\mathcal{Z}'$ forwards it if it is addressed to the instance of π. Otherwise, $\mathcal{Z}'$ ignores this message. When $\mathcal{Z}$ invokes and instance of $\mathrm{IDEAL}(\mathcal{F}^{\mathcal{O}})$, $\mathcal{Z}'$ provides a special invocation input to its challenge protocol instructing it to invoke an instance of λ using the same SID and PIDs as the instance of $\mathrm{IDEAL}(\mathcal{F}^{\mathcal{O}})$ invoked by $\mathcal{Z}$. $\mathcal{Z}'$ internally simulates each instance of $\mathrm{IDEAL}(\mathcal{F}^{\mathcal{O}})$ invoked by $\mathcal{Z}$ using the Imaginary Angel Γ. When $\mathcal{Z}$ sends an instruction to corrupt a party, $\mathcal{Z}'$ instucts the adversary to corrupt the party with the same identity. Messages coming from the adversary are forwarded by $\mathcal{Z}'$ to $\mathcal{Z}$. Outputs coming from an instance of λ are ignored by $\mathcal{Z}'$. Finally, $\mathcal{Z}'$ outputs whatever $\mathcal{Z}$ outputs.

Let $\mathcal{S} = \mathcal{D}$. It is easy to see that $\mathrm{Exec}_{\mathsf{UC}}(\rho^{\pi}, \mathcal{D}, \mathcal{Z}'^{\Gamma}) = \mathrm{Exec}\big(\pi, \mathcal{D}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]\big)$ and $\mathrm{Exec}_{\mathsf{UC}}(\rho^{\mathcal{F}}, \mathcal{S}_{\mathcal{D}}^{\Gamma}, \mathcal{Z}'^{\Gamma}) = \mathrm{Exec}\big(\mathcal{F}^{\mathcal{O}}, \mathcal{S}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]\big)$ holds (note that since Γ

is normal it behaves the same way in each instance of $\mathcal{O}$ as in the interaction with $\mathcal{Z}'$). Therefore, $\mathrm{Exec}\big(\pi, \mathcal{D}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]\big) \overset{c}{\equiv} \mathrm{Exec}\big(\mathcal{F}^{\mathcal{O}}, \mathcal{S}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}]\big)$. The statement now follows from the completeness of the dummy adversary (Proposition 4.8).

2. Let $\widetilde{\rho}$ be a string commitment protocol such that $\widetilde{\rho} \geq_{\mathcal{F}^{\mathcal{O}}_{\mathsf{stcom}}} \mathcal{F}^{\mathcal{O}}_{\mathsf{stcom}}$ and $\mathcal{F}^{\mathcal{O}}_{\mathsf{stcom}}$ is polynomially simulatable (cf. Section 2.4.3 for a definition of the ideal string commitment functionality $\mathcal{F}_{\mathsf{stcom}}$). One can find such a protocol and shielded oracle by taking the Angel-based protocol in [CLP10], which is based on one-way functions, and applying part 1 of this theorem and Lemma 4.18.[11]

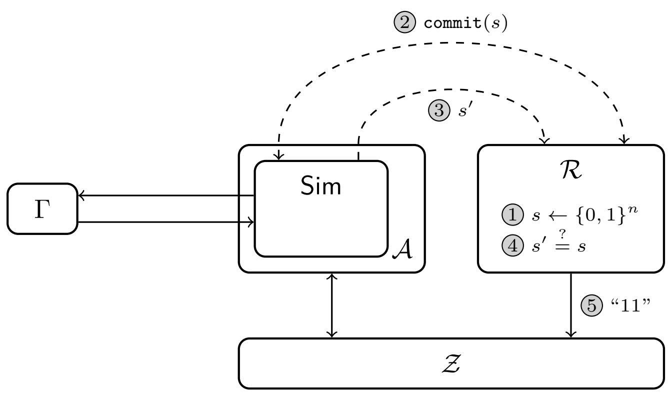
   Define the protocol ρ to be identical to $\widetilde{\rho}$ except for the following instruction:

   Before the actual commit phase begins, the receiver $\mathcal{R}$ chooses a bitstring $s \in \{0,1\}^n$ uniformly at random (n is the security parameter) and commits to s to the sender by running the program of the honest sender in $\widetilde{\rho}$ with the PID of the sender and the same SID as the protocol. The sender replies with, say, an all-1 string $(1,\dots,1) \in \{0,1\}^n$. The receiver $\mathcal{R}$ then checks if the bitstring he received from the sender equals s. If yes, $\mathcal{R}$ outputs "11" (2-bit string). Otherwise, the protocol parties execute the protocol $\widetilde{\rho}$.

   It holds that $\rho \geq_{\mathcal{F}^{\mathcal{O}}_{\mathsf{stcom}}} \mathcal{F}^{\mathcal{O}}_{\mathsf{stcom}}$. This follows from the following argument: First, it holds that $\rho \geq_{\mathcal{F}^{\mathcal{O}}_{\mathsf{stcom}}} \widetilde{\rho}$. This follows from the fact that an environment can only distinguish ρ and $\widetilde{\rho}$ if it guesses the string s correctly. However, the probability that an $\mathcal{F}^{\mathcal{O}}_{\mathsf{stcom}}$-augmented environment guesses s correctly is negligible. This is because since $\mathcal{F}^{\mathcal{O}}_{\mathsf{stcom}}$ is polynomially simulatable, every $\mathcal{F}^{\mathcal{O}}_{\mathsf{stcom}}$-augmented environment can be replaced by a PPT environment and PPT environments can guess s correctly only with negligible probability. In order to proof the latter claim, assume there is a PPT environment $\mathcal{Z}$ that guesses s correctly with non-negligible probability. Construct an adversary $\mathcal{A}$ interacting with $\widetilde{\rho}$ as follows: $\mathcal{A}$ corrupts the receiver. $\mathcal{A}$ internally simulates $\mathcal{Z}$ and forwards all messages from the sender to $\mathcal{Z}$ and vice versa. When $\mathcal{Z}$ outputs its guess, $\mathcal{A}$ forwards this guess to his own environment, which checks if the bitstring it received equals the input to the sender. By construction, $\mathcal{A}$'s output is correct with non-negligible probability. However, this cannot be simulated in the ideal model experiment, contradicting $\widetilde{\rho} \geq_{\mathcal{F}^{\mathcal{O}}_{\mathsf{stcom}}} \mathcal{F}^{\mathcal{O}}_{\mathsf{stcom}}$. Hence, $\rho \geq_{\mathcal{F}^{\mathcal{O}}_{\mathsf{stcom}}} \widetilde{\rho}$ holds and therefore $\rho \geq_{\mathcal{F}^{\mathcal{O}}_{\mathsf{stcom}}} \mathcal{F}^{\mathcal{O}}_{\mathsf{stcom}}$ also holds by the transitivity of $\mathcal{F}^{\mathcal{O}}_{\mathsf{stcom}}$-emulation (Proposition 4.9).

   Now assume for the sake of contradiction that there exists an Imaginary Angel Γ s. t. $\rho \geq_{\Gamma\text{-Angel}} \mathcal{F}_{\mathsf{stcom}}$ holds. Then there exists a simulator Sim^{Γ} for the dummy adversary that must be able to extract commitments (generated by running the program of the honest sender with the PID of the sender and the SID of the protocol) with overwhelming probability. Consider the following adversary $\mathcal{A}^{\Gamma}$. $\mathcal{A}^{\Gamma}$ corrupts the sender. $\mathcal{A}^{\Gamma}$ internally simulates a copy of the simulator Sim^{Γ} and forwards the commitment messages to the bitstring s between the receiver $\mathcal{R}$ and Sim^{Γ}. $\mathcal{A}^{\Gamma}$ then sends the output s' of Sim^{Γ} to the receiver as his guess. By construction, $\mathcal{A}^{\Gamma}$'s guess is correct

[11] One could also use the commitment scheme in Corollary 4.28 to obtain a separating example. However, this scheme is based on one-way permutations.



ⓝ	Step n
$\dashrightarrow$	authenticated communication between parties
commit(·)	Protocol messages of the base commitment $\widetilde{\rho}$

Figure 4.6: The adversary $\mathcal{A}^{\Gamma}$ in the proof of the second part of Theorem 4.21.

with overwhelming probability, forcing the receiver to output "11" in the real model experiment with overwhelming probability. This cannot be simulated in the ideal model experiment, however. We have thus reached a contradiction. (For an illustration, see Fig. 4.6).

□

Theorem 4.21 raises the question if it is possible to construct secure protocols with "interesting properties" in our framework that are not (known to be) secure in the Angel-based setting. We will answer this question in the affirmative and present a modular construction of a general MPC protocol in the plain model that is constant-round (and black-box) and based only on standard polynomial-time hardness assumptions (Theorem 4.39).

We would like to briefly note that by Theorem 4.21 we can already conclude that we can realize every standard well-formed[12] functionality in our framework in the plain model by importing the results of [CLP10].

Proposition 4.22 (General MPC in the Plain Model). *Assume the existence of enhanced trapdoor permutations. Then, for every standard well-formed functionality $\mathcal{F}$, there exists a shielded oracle $\mathcal{O}$ and a protocol ρ in the plain model such that*

$$\rho \underset{\mathcal{F}^{\mathcal{O}}}{\geq} \mathcal{F}^{\mathcal{O}}$$

4.4 A Constant-Round Commitment Scheme

In this section, we will construct a constant-round (bit) commitment scheme that is secure in our framework. We note that we assume authenticated communication and therefore implicitly work in the $\mathcal{F}_{\text{auth}}$-hybrid model (cf. Definition 2.23).

[12]Cf. Section 2.4.2 for a definition of standard well-formed functionalities

4.4.1 Construction and Security Proof

Let $\langle C, R\rangle$ be a tag-based commitment scheme[13] that we will use as a building block for our bit commitment scheme Π later. We require $\langle C, R\rangle$ to be *immediately committing* as in the following definition.

Definition 4.23 (Immediately Committing). *A commitment scheme $\langle C, R\rangle$ is called* immediately committing *if the first message in the protocol comes from the sender and already perfectly determines the value committed to.*

The above definition implies that the commitment scheme is perfectly binding and super-polynomially extractable, i. e., given a transcript, an extractor can find the unique committed value through exhaustive search.

In the description of our commitment scheme Π, we use the following notation: Let $s = ((s_{i,b})) \in \{0,1\}^{2n}$ for $i \in [n]$ and $b \in \{0,1\}$ be a $2n$-tuple of bits. For an n-bit string $I = b_1 \cdots b_n$, we define $s_I := (s_{1,b_1}, \ldots, s_{n,b_n})$. Thus I specifies a selection of n of the $s_{i,b}$, where one of these is selected from each pair $s_{i,0}, s_{i,1}$.

Construction 1. *The bit commitment scheme Π is defined as follows. Whenever the basic commitment scheme $\langle C, R\rangle$ is used, the committing party uses its pid and the sid as its tag. Let $m \in \{0,1\}$*

- Commit(m)*:*
  - R*: Choose a random n-bit string I and commit to I using $\langle C, R\rangle$*
  - S*: Pick n random bits $s_{i,0}$ and compute $s_{i,1} = s_{i,0} \oplus m$ for all $i \in [n]$.*
  - S *and* R *run $2n$ sessions of $\langle C, R\rangle$ in parallel in which* S *commits to the s_{i,b_i} $(i \in [n], b_i \in \{0,1\})$.*
- Unveil*:*
  - S*: Send all $s_{i,b_i} \in \{0,1\}$ $(i \in [n], b_i \in \{0,1\})$ to* R.
  - R*: Check if $s_{1,0} \oplus s_{1,1} = \ldots = s_{n,0} \oplus s_{n,1}$. If this holds, unveil the string I to* S.
  - S*: If* R *unveiled the string correctly, then unveil all s_I.*
  - R*: Check if* S *unveiled correctly. If yes, let $s'_1, \ldots, s'_n$ be the unveiled values. Check if $s'_i = s_{i,b_i}$ for all $i \in [n]$. If so, output $m := s_{1,0} \oplus s_{1,1}$.*

The above construction is reminiscent of [DS13] who presented a compiler that transforms any ideal straight-line extractable commitment scheme into an extractable and equivocal commitment scheme.

Note that if an adversary is able to learn the index set I in the commit phase then he can easily open the commitment to an arbitrary message m' by sending "fake" shares $t_{i,b}$, such that $t_I = s_I$, and $t_{\neg I} = s_I \oplus (m', \ldots, m')$. (Here $\oplus$ is interpreted element-wise.) Hence Π is equivocal for super-polynomial machines.

We claim that this protocol securely realizes $\mathcal{F}^{\mathcal{O}}_{\mathrm{com}}$ for a certain shielded oracle $\mathcal{O}$. We first describe $\mathcal{O}$, before we move to the theorem.

Construction 2. *We define the actions of the shielded oracle $\mathcal{O}$ as follows:*[14]

[13]Cf. Section 2.3.2 for a definition of tag-based commitment schemes.

[14]For ease of notation, we drop the prefixes `output-to-fnct` and `output-to-adv` in the messages output by $\mathcal{O}$.

If ***the sender is corrupted***

- *$\mathcal{O}$ chooses a random n-bit string I, and commits to the string I to the adversary $\mathcal{A}$ using $\langle C, R\rangle$.*
- *$\mathcal{O}$ acts as honest receiver in $2n$ sessions of $\langle C, R\rangle$ in parallel. After these sessions have completed, $\mathcal{O}$ extracts each instance of $\langle C, R\rangle$, obtaining the shares ($s_{i,b}$ for $i \in [n]$) and $b \in \{0,1\}$. (If a commitment cannot be extracted, the corresponding share is set to $\perp$)*
- *$\mathcal{O}$ computes $m_i := s_{i,0} \oplus s_{i,1}$ for all $i \in [n]$. (Indices i where one or both of the $s_{i,b}$ is $\perp$ are ignored.) Let $m \in \{0,1\}$ be the most frequently occurring m_i. (If there are multiple m_i occurring with the highest frequency, m chooses $m = 0$). $\mathcal{O}$ relays (Commit, m) to $\mathcal{F}_{com}$*
- *When $\mathcal{A}$ sends shares $s'_{1,0}, s'_{1,1}, \ldots, s'_{n,0}, s'_{n,1}$ in the unveil phase of Π, $\mathcal{O}$ acts as an honest receiver, unveiling I.*
- *Finally, if $\mathcal{A}$'s unveil is accepting, $\mathcal{O}$ instructs $\mathcal{F}_{com}$ to unveil the message.*

If ***the receiver is corrupted***

- *$\mathcal{O}$ acts as the sender in an execution of Π, engaging in a commit session of $\langle C, R\rangle$ with the adversary. If the adversary's commitment is accepting, $\mathcal{O}$ extracts this instance of $\langle C, R\rangle$ obtaining a string I (If parts of this string cannot be extracted they are set to $\perp$).*
- *$\mathcal{O}$ picks n random bits $s_{i,0}$, and lets $s_{i,1} = s_{i,0}$ for all $i \in [n]$, as if it were honestly committing to $m = 0$. Next, it runs $2n$ instances of Π in parallel, committing to the $s_{i,b}$.*
- *In the unveil phase, when $\mathcal{O}$ learns the message m, it computes "fake" shares $t_{i,b}$ as follows: $t_I = s_I$ and $t_{\neg I} = s_{\neg I} \oplus (m, \ldots, m)$ ($\oplus$ is interpreted element-wise.). $\mathcal{O}$ sends these shares $t_{i,b}$ to the adversary.*
- *$\mathcal{O}$ acts as the honest sender in the unveil phase of Π. If $\mathcal{A}$'s unveil of I is accepting, then $\mathcal{O}$ honestly executes the unveil phase for all bit shares t_I. (Otherwise, $\mathcal{O}$ outputs nothing and ignores all further inputs.)*

If ***no parties are corrupted****, $\mathcal{O}$ simulates an honest execution of protocol* Π *on input* 0*, forwarding all messages to the adversary. Since $\mathcal{O}$ knows the index string I (because $\mathcal{O}$ has created it itself) it can create fake shares just like in the case of a corrupted receiver.*

If ***both parties are corrupted****, $\mathcal{O}$ just executes the dummy adversary $\mathcal{D}$ internally. (Note that $\mathcal{Z}$ only interacts with $\mathcal{D}$ in the real experiment if both parties are corrupted).*

This concludes the description of the shielded oracle $\mathcal{O}$. Observe that $\mathcal{O}$ can be implemented in super-polynomial time. Also note that in the case of *both or no* party being corrupted, $\mathcal{O}$ can be implemented in polynomial time.

Before we state and prove our theorem, we define the following security notion that will be used in the proof:

Definition 4.24 (Strong $\mathcal{O}_{\mathsf{pcca}}$-one-way Hiding)**.** *Let $\langle C, R\rangle$ be a tag-based commitment scheme. Let $\mathsf{Exp}^{\mathsf{stow}}_{\langle C,R\rangle,\mathcal{A}(z)}(n)$ denote the output of the following probabilistic experiment: On input $1^n, z$, $\mathcal{A}$ selects a tag* tag *and sends that tag to the experiment. The experiment then picks a random string $I = b_1 \cdots b_n$ and commits to I using $\langle C, R\rangle$ and the tag* tag *selected by the adversary. $\mathcal{A}$ then sends a vector $(a_1, \ldots, a_n)$ to the experiment, where $a_i \in \{0, 1, \perp\}$. Let $M = \{l \mid a_l \neq \perp\}$. The output of the experiment is* 1 *if* card$(M) \geq n/2$ *and $a_l = b_l$ for all $l \in M$, and* 0 *otherwise. During the experiment, $\mathcal{A}$ has access to the parallel CCA oracle $\mathcal{O}_{\mathsf{pcca}}$ of $\langle C, R\rangle$.*

An adversary $\mathcal{A}$ is called valid *if he does not query the parallel CCA oracle on tags that are equal to the challenge tag* tag *during the experiment.*

$\langle C, R\rangle$ is called strong $\mathcal{O}_{\mathsf{pcca}}$-one-way hiding *if for every valid PPT adversary $\mathcal{A}$ there exists a negligible* negl *such that for all $n \in \mathbb{N}, z \in \{0,1\}^*$ it holds that*

$$\mathsf{Pr}[\mathsf{Exp}^{\mathsf{stow}}_{\langle C,R\rangle,\mathcal{A}(z)}(n) = 1] \leq \mathsf{negl}(n)$$

We have the following easy lemma.

Lemma 4.25. *If a commitment scheme $\langle C, R\rangle$ is parallel CCA-secure, then it is also strong $\mathcal{O}_{\mathsf{pcca}}$-one-way hiding*

Proof idea. Replace the challenge commitment (which is a commitment to a random string I) with a commitment to 0^n using the parallel CCA security of $\langle C, R\rangle$. This way, the view of the adversary $\mathcal{A}$ becomes independent of I. $\mathcal{A}$ can therefore only guess. Since $\mathcal{A}$ has to guess at least $n/2$ bits of I, $\mathcal{A}$ will only win with negligible probability. □

We are now ready to state the theorem:

Theorem 4.26. *Assume that $\langle C, R\rangle$ is parallel CCA-secure*[15] *and immediately committing. Then $\Pi \geq_{\mathcal{F}^{\mathcal{O}}_{com}} \mathcal{F}^{\mathcal{O}}_{com}$, where Π is as defined in Construction 1 and $\mathcal{O}$ is the shielded oracle as defined in Construction 2.*

Proof. By Proposition 4.8 it suffices to find a simulator for the dummy adversary. By construction of $\mathcal{O}$, the simulator in the ideal experiment can be chosen to be identical to the dummy adversary.

The main idea of the proof is to consider a sequence of hybrid experiments for a PPT environment $\mathcal{Z}$ that may externally invoke polynomially instances of IDEAL$(\mathcal{F}^{\mathcal{O}})$ and iteratively replace all those instances by the real protocol Π. This is done in a specific order utilizing the fact that the super-polynomial computations of $\mathcal{O}$ are hidden away and thus the replacements are unnoticeable by $\mathcal{Z}$, or otherwise we would obtain a PPT adversary against the parallel CCA security of $\langle C, R\rangle$.

Step 1. Let $\mathcal{Z}$ be a (PPT) environment that externally invokes polynomially many instances of IDEAL$(\mathcal{F}^{\mathcal{O}})$. We denote the output of $\mathcal{Z}$ by the random variable $\mathrm{Exec}\big(\mathcal{F}^{\mathcal{O}}_{\mathrm{com}}, \mathcal{Z}\big)$. Furthermore, we denote by $\mathrm{Exec}\big(\Pi, \mathcal{Z}\big)$ the output of $\mathcal{Z}$ if all instances of IDEAL$(\mathcal{F}^{\mathcal{O}})$ are replaced by instances of the real protocol Π. We show that the following holds

$$\mathrm{Exec}\big(\mathcal{F}^{\mathcal{O}}_{\mathrm{com}}, \mathcal{Z}\big) \overset{\mathrm{c}}{\equiv} \mathrm{Exec}\big(\Pi, \mathcal{Z}\big)$$

[15] Cf. Definition 3.11 for a definition of this notion

By a standard averaging argument, we can fix some random coins r for $\mathcal{Z}$. Thus we can assume henceforth that $\mathcal{Z}$ is deterministic.

We call instances of IDEAL($\mathcal{F}^{\mathcal{O}}$) or Π where the sender or receiver is corrupted *sender sessions* or *receiver sessions*, respectively. Since instances of IDEAL($\mathcal{F}^{\mathcal{O}}$) can be implemented in polynomial time in the cases where both or no party is corrupted, they can be treated as part of the environment in these cases. Hence, we only need to consider environments that only invoke ideal sender sessions and ideal receiver sessions.

We say a *discrepancy* occurred if in any ideal sender session the oracle $\mathcal{O}$ extracts a value m, but $\mathcal{Z}$ later correctly unveils a value $m' \neq m$. First notice that unless a discrepancy occurs, the output of an ideal sender session is identically distributed to the output of the real protocol Π.

We will now distinguish two cases:

1. The probability that $\mathcal{Z}$ causes a discrepancy is negligible.
2. The probability that $\mathcal{Z}$ causes a discrepancy is non-negligible.

Case 1: We replace all ideal sender sessions with instances of Π, incurring only a negligible statistical distance. We are left with a hybrid experiment in which only the receiver sessions are still ideal. We will now iteratively replace ideal receiver sessions with the real protocol, beginning with the *last* receiver session that is started.

Let q be an upper bound on the number of ideal receiver sessions that $\mathcal{Z}$ invokes. Define hybrids $\mathsf{H}_0, \ldots, \mathsf{H}_q$ as follows. Hybrid H_i is the experiment where the first i receiver sessions are ideal and the remaining $q - i$ ideal receiver sessions are replaced by instances of Π. The output of the experiment H_i is the output of $\mathcal{Z}$. Clearly, H_q is identical to the experiment where all receiver sessions are ideal, whereas H_0 is the experiment where all receiver sessions are real. Let $P_i = \mathsf{Pr}[\mathsf{H}_i = 1]$ denote the probability that $\mathcal{Z}$ outputs 1 in the hybrid experiment H_i. Assume now that $\epsilon := |P_0 - P_q|$ is non-negligible, i. e., $\mathcal{Z}$ has non-negligible advantage ϵ in distinguishing H_0 from H_q. We will now construct an adversary $\mathcal{A}$ that breaks the parallel CCA security of Π.

By the averaging principle, there must exist an index $i^* \in [q]$ such that $|P_{i^*-1} - P_{i^*}| \geq \epsilon/q$. By a standard coin-fixing argument, we can now fix the coins that are used in the first $i^* - 1$ receiver sessions up until the point where the $i^{*\text{th}}$ receiver session starts, while maintaining $\mathcal{Z}$'s distinguishing advantage.[16] Since we fixed $\mathcal{Z}$'s coins before, the experiment is now deterministic until the start of session i^*. In particular, the first message of $\mathcal{Z}$ in session i^*, which is the first message of a commitment on a bitstring I using $\langle C, R\rangle$, is computed deterministically.

We can now construct the non-uniform adversary $\mathcal{A}$ against the parallel CCA security of $\langle C, R\rangle$. As a non-uniform advice, $\mathcal{A}$ receives a complete trace of all messages sent up until the point where the $i^{*\text{th}}$ receiver session starts. This

[16]Note that "*fixing the coins up until the point where event E occurs*" means the following: Let M be the (non-uniform) turing machine that internally simulates the entire execution experiment given the security parameter 1^n, the environment's input z and two random tapes $r_1, r_2 \in \{0,1\}^L$, where $L = poly(n)$ is an upper bound on the number of random coins used in the experiment. $M(1^n, z, r_1, r_2)$ simulates the execution experiment using the random tape r_1 up until the point where event E occurs and uses r_2 afterwards. "Fixing the coins up until the point where event E occurs" now means that the first random tape r_1 is fixed.

includes all bit strings $I_1, \dots, I_{i^*}$ to which $\mathcal{Z}$ committed to in all receiver sessions $1, \dots, i^*$. Note that all messages sent up until the point where session $i^{*\text{th}}$ starts come from a deterministic process, and the I_i are uniquely determined by the first messages of each session i since $\langle C, R\rangle$ is immediately committing.

$\mathcal{A}$ now proceeds as follows. $\mathcal{A}$ internally simulates $\mathcal{Z}$ and all sessions invoked by $\mathcal{Z}$. This simulation can be done in *polynomial time* since all ideal sender sessions and the ideal receiver sessions $i^* + 1$ through q have been replaced by instances of Π, and $\mathcal{A}$ knows the index strings I_i that are used in the ideal receiver sessions 1 through i^*.

Let m^* be the message that $\mathcal{Z}$ chooses as input for the sender in the receiver session i^*. $\mathcal{A}$ reads $I \overset{\text{def}}{=} I_{i^*}$ from its non-uniform advice and samples a tuple s_I of n random strings. It then computes $s_{\neg I} = s_I \oplus (m^*, \dots, m^*)$ and $s'_{\neg I} = s_I$ for all $i \in [n]$. $\mathcal{A}$ sends the messages $(s_{\neg I}, s'_{\neg I})$ to the parallel CCA security experiment. In order to simulate the $i^{*\text{th}}$ receiver session in its internal simulation, $\mathcal{A}$ forwards all the messages between the experiment and $\mathcal{Z}$ and simultaneously commits honestly to all values s_I to $\mathcal{Z}$. When $\mathcal{Z}$ requires that the commitments to all s_I be opened, $\mathcal{A}$ honestly unveils these. When $\mathcal{Z}$ terminates, $\mathcal{A}$ outputs whatever $\mathcal{Z}$ outputs in the experiment. This concludes the description of $\mathcal{A}$.

It holds that if the parallel CCA security experiment picks the messages $s'_{\neg I}$, $\mathcal{Z}$ obtains a commitment to the all-zero string in $\mathcal{A}$'s simulation. Therefore, in this case the view of $\mathcal{Z}$ is distributed identically to the view inside the hybrid H_{i^*}. If the parallel CCA security experiment picks the messages $s_{\neg I}$, $\mathcal{Z}$ obtains a commitment to the message m which is identical to the view of $\mathcal{Z}$ inside the hybrid H_{i^*-1}. Therefore, it follows that

$$\begin{aligned}
&\left|\text{Output}^{\mathsf{pcca}}_{\mathsf{Com},\mathcal{A}}(0) - \text{Output}^{\mathsf{pcca}}_{\mathsf{Com},\mathcal{A}}(1)\right| \\
&= \left|\mathsf{Pr}[\mathsf{H}_{i^*} = 1] - \mathsf{Pr}[\mathsf{H}_{i^*-1} = 1]\right| \\
&= |P_{i^*} - P_{i^*-1}| \\
&\geq \epsilon/q,
\end{aligned}$$

Hence, $\mathcal{A}$ breaks the parallel CCA security of $\langle C, R\rangle$ (note that in this case $\mathcal{A}$ does not need the parallel CCA oracle.)

Case 2: We now turn to case 2. First note that if there exists an environment that causes a discrepancy with non-negligible probabilty than there also exists an environment that invokes, apart from polynomially many ideal receiver sessions, *exactly one* ideal sender session and also causes a discrepancy with non-negligible probabilty. This is because if a general environment $\mathcal{Z}$ causes a discrepancy with non-negligible probability, then there exists a session j^* in which a discrepancy happens *for the first time.* An environment $\mathcal{Z}'$ that invokes only one ideal sender session can then internally run $\mathcal{Z}$, guess j^* and simulate all the other sender sessions with the real protocol. It holds that $\mathcal{Z}'$ also causes a discrepancy with non-negligible probability.

So we henceforth assume that there exists an environment $\mathcal{Z}$ that causes a discrepancy with non-negligible probabilty and invokes at most q ideal receiver sessions and only one ideal sender session. In what follows, we will replace all ideal receiver sessions with instances of the real protocol Π using the same strategy as in case 1. Define the hybrids $\mathsf{H}_0, \dots, \mathsf{H}_q$ as in case 1 except that now $\mathcal{Z}$ can additionally invoke exactly one ideal sender session in all these hybrids. Clearly, H_q is identical to the experiment where all receiver sessions are ideal, whereas H_0 is the experiment where all receiver sessions are real. Let $P_i = \mathsf{Pr}[\mathsf{H}_i = 1]$ again.

Assume that $\mathcal{Z}$ can distinguish between H_0 and H_q with non-negligible advantage ϵ. Then there exists an index i^* such that $|P_{i^*-1} - P_{i^*}| \geq \epsilon/q$. We can now fix the coins that are used in the first $i^* - 1$ receiver sessions and the one ideal sender session up until the point where the $i^{*\text{th}}$ receiver session starts, while maintaining $\mathcal{Z}$'s distinguishing advantage.

We will construct a non-uniform adversary $\mathcal{A}'$ that breaks the parallel CCA security of $\langle C, R\rangle$. As in case 1, $\mathcal{A}'$ receives as a non-uniform advice a complete trace of all messages sent up until the point where the $i^{*\text{th}}$ receiver session starts which also includes all index strings I_i to which $\mathcal{Z}$ committed in all receiver sessions up to and including the receiver session i^*, and possibly the shares to which $\mathcal{Z}$ committed in the only ideal sender session.

$\mathcal{A}'$ now proceeds the same way as in case 1. It internally runs $\mathcal{Z}$ and simulates either hybrid H_{i^*-1} or H_{i^*} for $\mathcal{Z}$ by embedding the challenge of the parallel CCA security experiment into the simulated receiver session i^*. The adversary $\mathcal{A}'$ simulates all (ideal) receiver sessions $i \leq i^*$ with the help of his advice, and all receiver sessions $i > i^*$ by running instances of the real protocol Π. If $\mathcal{Z}$ has already started to commit to the shares in the only ideal sender session then (by definition) these shares are also part of $\mathcal{A}'$'s advice and $\mathcal{A}'$ can simulate the ideal sender session using its advice. (Note that $\langle C, R\rangle$ is immediately committing, hence the first message of (the parallel executions of) $\langle C, R\rangle$ uniquely determines the shares). If $\mathcal{Z}$ has not yet started to commit to the shares in the ideal sender session, then $\mathcal{A}'$ can use its parallel CCA oracle to extract them by forwarding the corresponding messages between the oracle and $\mathcal{Z}$ (note that the tag in $\mathcal{A}$'s challenge is different from the tag in his oracle query since the sessions invoked by $\mathcal{Z}$ have unique SIDs). Finally, $\mathcal{A}'$ outputs whatever $\mathcal{Z}$ outputs.

The analysis of $\mathcal{A}'$ is the same as in case 1 and we end up with the conclusion that $\mathcal{A}'$ breaks the parallel CCA security of protocol $\langle C, R\rangle$.

Since a discrepancy can be detected in polynomial-time, the above argument allows us to reduce to environments that cause a discrepancy with non-negligible probability and only invoke exactly one ideal sender session. Let $\mathcal{Z}$ be such an environment.

We will now construct an adversary $\mathcal{A}''$ against the strong $\mathcal{O}_{\mathsf{pcca}}$-one-way hiding security of $\langle C, R\rangle$ (cf. Definition 4.24). $\mathcal{A}''$ proceeds as follows: $\mathcal{A}''$ forwards the messages between the experiment and $\mathcal{Z}$ for the commitment to a random bitstring I that $\mathcal{Z}$ expects from the oracle $\mathcal{O}$ in the ideal sender session. When $\mathcal{Z}$ sends the commitments on the shares $s_{i,b}$, $\mathcal{A}''$ forwards them to its parallel CCA oracle, thus learning the values $s_{i,b}$ that $\mathcal{Z}$ committed to. $\mathcal{A}$ can now simulate the oracle $\mathcal{O}$ and reconstruct the message m defined by these shares (by defining m to be the most frequent value that occurs in $\{s_{i,0} \oplus s_{i,1}\}_{i\in[n]}$). When $\mathcal{Z}$ sends the shares $s'_{i,b}$ in the unveil phase, $\mathcal{A}''$ compares them to the extracted shares $s_{i,b}$ and defines the vector $(a_1, \ldots, a_n)$ as

$$a_l := \begin{cases} b_l & \text{if } \exists\, b_l \in \{0,1\} : s_{l,b_l} = s'_{l,b_l} \wedge s_{l,\neg b_l} \neq s'_{l,\neg b_l} \\ \perp & \text{else (if no such } b_l \text{ exists)} \end{cases} \qquad (\star)$$

and sends $(a_1, \ldots, a_n)$ to the experiment.

We will now analyze $\mathcal{A}''$'s success probability. Let M be the set of indices l for which condition ($\star$) holds. If $\mathcal{Z}$ causes a discrepancy, then it holds that all tuples of shares $(s'_{i,0}, s'_{i,1})$ define the same message m' and this message m' is different from m, the most frequently occuring value of $\{s_{i,0} \oplus s_{i,1}\}_{i \in [n]}$. Thus $\mathrm{card}(M) \geq n/2$. Furthermore, for each $l \in M$, b_l must equal the lth bit of the bitstring to which $\mathcal{A}''$'s experiment has committed if $\mathcal{Z}$ causes a discrepancy.

Hence, $\mathcal{A}''$ wins with non-negligible probability in the strong $\mathcal{O}_{\mathsf{pcca}}$-one-way hiding experiment since $\mathcal{Z}$, by assumption, causes a discrepancy with non-negligible probability.

Step 2. We will now prove that for every $\mathcal{F}^{\mathcal{O}}$-augmented environment

$$\mathrm{Exec}\big(\Pi, \mathcal{D}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}_{\mathrm{com}}]\big) \overset{c}{\equiv} \mathrm{Exec}\big(\mathcal{F}^{\mathcal{O}}_{\mathrm{com}}, \mathcal{D}, \mathcal{Z}[\mathcal{F}^{\mathcal{O}}_{\mathrm{com}}]\big) \ .$$

If the *sender is corrupted* then the real and ideal experiment are statistically close. This follows from the fact that by step 1, case 2, an $\mathcal{F}^{\mathcal{O}}_{\mathrm{com}}$-augmented environment can cause a discrepancy only with negligible probability.

If the *receiver is corrupted* then by step 1 the real and ideal experiment are both indistinguishable to the experiment where all instances of $\mathrm{IDEAL}(\mathcal{F}^{\mathcal{O}})$ have been replaced by instances of the real prototol. Hence the outputs of the real and ideal experiment are indistinguishable by transitivity.

If *no party is corrupted* then one can first replace all ideal sender sessions and ideal receiver sessions with the real protocol using step 1, obtaining a polynomial time environment that does not invoke super-polynomial time entities. Then one can prove indistinguishability by using a very similar reduction to the parallel CCA security as in step 1, case 1.

If *both parties are corrupted* then the real and ideal experiment are identically distributed. □

Remark 4.27 (Separating Example between SPS Security and Shielded Oracles)**.** *One can obtain a protocol that is secure in the SPS security framework but not in our framework by assuming that the commitment scheme $\langle C, R\rangle$ in construction 1 is only computationally hiding, statistically binding and extractable (cf. Definition 2.10) but not non-malleable.*

Concretely, instantiating $\langle C, R\rangle$ with the extractable commitment scheme in [PW09] yields the desired result (note that the protocol in [PW09] is not a tag-based commitment scheme but can be trivially made into (a malleable) one where the parties simply ignore the tag). SPS security follows easily from some of the arguments in the above proof (in particular, the case of a corrupted sender follows from a similar argument as in the reduction to the strong $\mathcal{O}_{\mathsf{pcca}}$-one-way hiding security of $\langle C, R\rangle$ in step 1, case 2; the case of a corrupted receiver follows from a similar argument as in the reduction to the parallel CCA security of $\langle C, R\rangle$ in step 1, case 1). Furthermore, since an augmented environment interacting with this protocol can easily cause a discrepancy by simply corrupting the sender and forwarding the messages between the honest receiver of this protocol and an additionally invoked ideal receiver session, this protocol is not secure in our framework.

The underlying commitment scheme $\langle C, R\rangle$ can be instantiated with the 8-round construction in [Goy+14]. It is straightforward to see that this construction is parallel CCA-secure by using the extractor in its security proof. Deviating from

[Goy+14] (which used the two-round protocol from [Nao89] based on one-way functions), we instantiate the basic commitment scheme in this construction with the non-interactive perfectly binding scheme from [Blu81] (which is based on one-way permutations) because we require an immediately committing protocol. Since this instantiation is constant-round, we obtain the following result:

Corollary 4.28. *Assume the existence of one-way permutations. Then there exists a constant-round bit commitment protocol Π_{com} and a shielded oracle $\mathcal{O}$ such that $\Pi_{com} \geq_{\mathcal{F}^{\mathcal{O}}_{com}} \mathcal{F}^{\mathcal{O}}_{com}$.*

The protocol Π_{com} in Corollary 4.28 is non-black-box because the construction from [Goy+14] (instantiated as described above) is non-black-box. However, the only non-black-box part of the construction from[Goy+14] is a ZK proof for proving knowledge of committed values and that these values satisfy linear relations. As pointed out in [Goy+14], this can both be done making only black-box use of a homomorphic commitment scheme. Instantiating the construction from [Goy+14] with a (perfectly binding) homomorphic commitment scheme thus yields a fully black-box construction. Since we need an immediately committing scheme in the plain model for our protocol we let the sender (and not a trusted setup) generate the commitment key of the homomorphic commitment scheme. This can be done if the homomorphic commitment scheme is "verifiable". A verifiable homomorphic commitment scheme allows to non-interactively verify that a commitment key is well-formed (cf. Definition 2.9 for a formal definition). With this instantiation, we obtain the following corollary:

Corollary 4.29. *Assume the existence of verifiable perfectly binding homomorphic commitment schemes. Then there exists a constant-round* black-box *bit commitment protocol Π^{BB}_{com} and a shielded oracle $\mathcal{O}$ such that $\Pi^{BB}_{com} \geq_{\mathcal{F}^{\mathcal{O}}_{com}} \mathcal{F}^{\mathcal{O}}_{com}$.*

4.4.2 First Application: Constant-Round (Black-Box) ZK

As a first application of the commitment scheme constructed in the previous section, we present two constant-round zero-knowledge protocols in the plain model that are secure in our framework.[17]

By [CF01], there exists a constant-round UC-secure zero-knowledge protocol $\rho^{\mathcal{F}_{\text{com}}}$ in the $\mathcal{F}_{\text{com}}$-hybrid model. Since this protocol is *unconditionally secure*, one can find a shielded oracle $\mathcal{O}'$ such that $\rho^{\mathcal{F}^{\mathcal{O}}_{\text{com}}} \geq_{\mathcal{F}^{\mathcal{O}}_{\text{com}}, \mathcal{F}^{\mathcal{O}'}_{\text{ZK}}} \mathcal{F}^{\mathcal{O}'}_{\text{ZK}}$, where $\mathcal{O}$ is the shielded oracle as defined in Construction 2. $\mathcal{O}'$ is defined as the simulator for the (super-polynomial-time) adversary $\mathcal{D}^{\mathcal{O}}$. Combining Corollary 4.13 and Corollary 4.28, one obtains a (non-black-box) constant-round zero-knowledge protocol based on OWPs.

Corollary 4.30. *Assume the existence of one-way permutations. Then there exists a constant-round zero-knowledge protocol Π_{ZK} and a shielded oracle $\mathcal{O}$ such that $\Pi_{ZK} \geq_{\mathcal{F}^{\mathcal{O}}_{ZK}} \mathcal{F}^{\mathcal{O}}_{ZK}$.*

Furthermore, using the protocol [CF01] and combining Corollary 4.13 and Corollary 4.29, one obtains a constant-round and black-box ZK protocol based on verifiable perfectly binding homomorphic commitment schemes.

[17]Note that we will later construct two general MPC protocols that require stronger assumptions than the ZK protocols presented in this section (namely enhanced trapdoor permutations for the first construction and verifiable perfectly binding homomorphic commitment schemes plus PKE with oblivious public-key generation for the second construction.)

Corollary 4.31. *Assume the existence of verifiable perfectly binding homomorphic commitment schemes. Then there exists a constant-round black-box zero-knowledge protocol Π_{ZK}^{BB} and a shielded oracle $\mathcal{O}$ such that $\Pi_{ZK}^{BB} \geq_{\mathcal{F}_{ZK}^{\mathcal{O}}} \mathcal{F}_{ZK}^{\mathcal{O}}$.*

4.4.3 A Modular Composition Theorem

We show that we can plug the protocol Π from Construction 1 into a large class of UC-secure protocols in the $\mathcal{F}_{\text{com}}$-hybrid model in such a way that the composite protocol is secure in our framework. We first define Commit-Compute protocols and parallel-CCA-UC-emulation.

Definition 4.32 (Commit-Compute Protocols)**.** *Let $\rho^{\mathcal{F}_{com}}$ be a protocol in the $\mathcal{F}_{com}$-hybrid model. We call $\rho^{\mathcal{F}_{com}}$ a* commit-compute *protocol or* CC *protocol if it can be broken down into two phases: An initial* commit phase*, where the only communication allowed is sending messages to instances of $\mathcal{F}_{com}$. After the commit phase is over, a* compute phase *begins where sending messages to instances of $\mathcal{F}_{com}$ except for `Unveil`-messages is prohibited, but all other communication is allowed.*

Technical Remarks:

- Note that one cannot plug a commitment protocol π that is secure in our framework into an arbitrary UC-secure protocol and always obtain a protocol secure in our framework. Consider any protocol $\rho^{\mathcal{F}_{\text{com}}}$ that UC-realizes the coin toss functionality $\mathcal{F}_{\mathsf{ct}}$ in the $\mathcal{F}_{\text{com}}$-hybrid model (take, e.g., the coin flipping protocol by Blum [Blu81]). Construct a new protocol $\tilde{\rho}^{\mathcal{F}_{\text{com}}}$ out of $\rho^{\mathcal{F}_{\text{com}}}$ in the following way:

  $\tilde{\rho}^{\mathcal{F}_{\text{com}}}$ is identical to $\rho^{\mathcal{F}_{\text{com}}}$ except that in the beginning the receiver chooses n random bits $a_1, \dots, a_n$ and commits to these bits using π in a predetermined scheduling. The sender then commits to, say, an all-1 string by sending $(\texttt{commit}, 1)$ to n instances of $\mathcal{F}_{\text{com}}$. After the receiver has reveived an OK from all n instances of $\mathcal{F}_{\text{com}}$, he unveils all bits. The sender then unveils his bits. The receiver then checks if the bitstring $(b_1, \dots, b_n)$ it has received from the sender equals the bitstring $(a_1, \dots, a_n)$. If this holds, the receiver outputs "11" (2-bit string). Otherwise, both parties execute the protocol $\rho^{\mathcal{F}_{\text{com}}}$.

  Since π is hiding, it holds that $\tilde{\rho}^{\mathcal{F}_{\text{com}}} \geq_{\mathsf{UC}} \mathcal{F}_{\mathsf{ct}}$. However, there exists no shielded oracle $\mathcal{O}'$ such that $\tilde{\rho}^{\pi} \geq_{\mathcal{F}_{\mathsf{ct}}^{\mathcal{O}'}} \mathcal{F}_{\mathsf{ct}}^{\mathcal{O}'}$. This is because the adversary can simply forward the commitments to the a_i to instances of π, thereby forcing the receiver to output "11".
- We implicitly assume that at each activation during the commit phase each party *deterministically* decides if it commits or not during the current activation. This prevents a protocol party from transmitting the result $b = (b_1, \dots, b_n)$ of a local randomized computation by deciding to commit at the i-th activation only if the bit b_i equals, say, 1. In particular, it prevents the receiver from running a variation of the above coin flipping example by sending the commitments to the random bits $a_1, \dots, a_n$ this way.

Definition 4.33 (pCCA-UC-Emulation)**.** *We write $\rho \geq_{O_{\mathsf{pcca}}\text{-}\mathsf{UC}} \phi$ if a protocol ρ UC-emulates a protocol ϕ in the presence of PPT environments that may interact with the parallel CCA oracle $\mathcal{O}_{\mathsf{pcca}}$ of a tag-based commitment scheme $\langle C, R \rangle$.*

In the following, let Π be the protocol as in Construction 1 with an immediately committing and parallel CCA-secure commitment scheme $\langle C, R \rangle$ as building block. Let $\mathcal{O}_{\mathsf{pcca}}$ be the parallel CCA oracle of $\langle C, R \rangle$.

We are now ready to state the theorem:

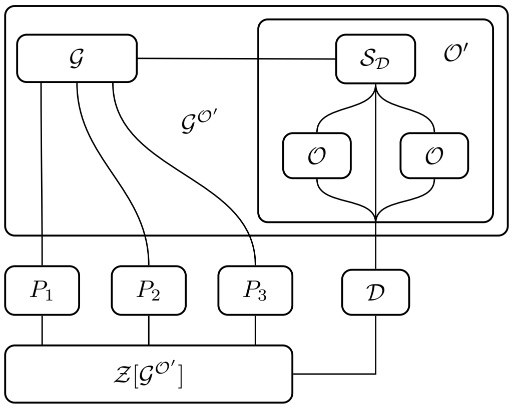

$\mathcal{G}$
$\mathcal{S}_\mathcal{D}$
$\mathcal{O}'$
$\mathcal{G}^{\mathcal{O}'}$
$\mathcal{O}$
$\mathcal{O}$
P_1
P_2
P_3
$\mathcal{D}$
$\mathcal{Z}[\mathcal{G}^{\mathcal{O}'}]$

Figure 4.7: The functionality $\mathcal{G}$ with composed oracle $\mathcal{O}'$

Theorem 4.34 (Modular Composition Theorem)**.** *Let $\rho^{\mathcal{F}_{com}}$ be a CC protocol and $\mathcal{G}$ a functionality. If $\rho^{\mathcal{F}_{com}} \geq_{O_{\mathsf{pcca}}\text{-}\mathsf{UC}} \mathcal{G}$ then there exists a shielded oracle $\mathcal{O}'$ such that*

$$\rho^{\Pi} \underset{\mathcal{G}^{\mathcal{O}'}}{\geq} \mathcal{G}^{\mathcal{O}'}$$

Proof. Since $\rho^{\mathcal{F}_{\text{com}}} \geq_{O_{\mathsf{pcca}}\text{-}\mathsf{UC}} \mathcal{G}$ there exists a dummy adversary simulator $\mathcal{S}_\mathcal{D}$. Let $\mathcal{O}$ be the shielded oracle from Construction 2, s. t. $\Pi \geq_{\mathcal{F}^{\mathcal{O}}_{\text{com}}} \mathcal{F}^{\mathcal{O}}_{\text{com}}$. We define the shielded oracle $\mathcal{O}'$ as follows. (For a graphical depiction see Fig. 4.7).

$\mathcal{O}'$ internally simulates multiple instances of $\mathcal{O}$ (one for each instance of $\mathcal{F}_{\text{com}}$ in ρ) and $\mathcal{S}_\mathcal{D}$, and forwards messages as follows.

- Messages from the adversary addressed to an instance of $\mathcal{F}_{\text{com}}$ are forwarded to the corresponding internal instance of $\mathcal{O}$.
- Messages from an internal instance of $\mathcal{O}$ to an instance of $\mathcal{F}_{\text{com}}$ are forwarded to the dummy adversary simulator $\mathcal{S}_\mathcal{D}$.
- Messages between $\mathcal{S}_\mathcal{D}$ and the functionality $\mathcal{G}$ are forwarded.
- Messages from the dummy adversary simulator $\mathcal{S}_\mathcal{D}$ addressed as coming from an instance of $\mathcal{F}_{\text{com}}$ are forwarded to the respective instance of $\mathcal{O}$.
- Messages from the dummy adversary simulator $\mathcal{S}_\mathcal{D}$ not addressed as coming from an instance of $\mathcal{F}_{\text{com}}$ are output to the adversary (without forwarding them to an internal instance of $\mathcal{O}$).

We claim that for this oracle $\rho^{\Pi} \geq_{\mathcal{G}^{\mathcal{O}'}} \mathcal{G}^{\mathcal{O}'}$ holds. By Proposition 4.8 it is sufficient to find a simulator for the dummy adversary. The simulator will be the dummy adversary in the ideal world.

Recall that we call instances of IDEAL($\mathcal{F}^{\mathcal{O}}$) or Π where the sender or receiver is corrupted *sender sessions* or *receiver sessions*, respectively. We denote by $\rho^{\Pi_{\text{S}},\mathcal{F}^{\mathcal{O}}_{\text{com}}}$ the protocol $\rho^{\mathcal{F}^{\mathcal{O}}_{\text{com}}}$ where all ideal sender sessions have been replaced by the real protocol.

Let $\mathcal{Z}$ be a (PPT) environment invoking polynomially many instances of (the CC protocol) $\rho^{\Pi_{\text{S}},\mathcal{F}^{\mathcal{O}}_{\text{com}}}$. We denote the output of $\mathcal{Z}$ by the random variable

$\mathrm{Exec}(\rho^{\Pi_{\mathrm{S}},\mathcal{F}_{\mathrm{com}}^{\mathcal{O}}}, \mathcal{Z})$. Furthermore, we denote by $\mathrm{Exec}(\mathcal{G}^{\mathcal{O}'}, \mathcal{Z})$ the output of $\mathcal{Z}$ if all instances of $\rho^{\Pi_{\mathrm{S}},\mathcal{F}_{\mathrm{com}}^{\mathcal{O}}}$ are replaced by instances of $\mathcal{G}^{\mathcal{O}'}$.

In the following hybrid argument, we will have to globally order the protocol instances invoked by $\mathcal{Z}$ according to when their commit phase is over and (adaptively) invoke instances of $\rho^{\Pi_{\mathrm{S}},\mathcal{F}_{\mathrm{com}}^{\mathcal{O}}}$, $\rho^{\mathcal{F}_{\mathrm{com}}^{\mathcal{O}}}$ or $\mathcal{G}^{\mathcal{O}'}$ based on this order. Since the message scheduling may be random, however, this order is not determined a-priori. We will therefore have the experiment in the hybrids implement the commit phases of all invoked protocols "obliviously", i. e., interact with the environment by running the programs of the shielded oracles and store the inputs of the honest parties without following their instructions in the commit phases. (Note that the only communication that is *visible* to the environment in the commit phase is its interaction with the shielded oracles or the receiver in an instance of Π_{S}. The latter interaction is identical to an interaction with the shielded oracle in a sender session. Furthermore, the inputs of the honest parties have no effect on the messages the shielded oracles output to the environment in a commit phase.) Once the commit phases of an instance of $\rho^{\Pi_{\mathrm{S}},\mathcal{F}_{\mathrm{com}}^{\mathcal{O}}}$ has ended, the experiment in the hybrids will invoke an instance of $\rho^{\Pi_{\mathrm{S}},\mathcal{F}_{\mathrm{com}}^{\mathcal{O}}}$, $\rho^{\mathcal{F}_{\mathrm{com}}^{\mathcal{O}}}$ or $\mathcal{G}^{\mathcal{O}'}$ depending on the position within the global order of sessions. The experiment will then invoke the honest parties with their respective inputs and follow their instructions (it will also invoke the simulator $\mathcal{S}_{\mathcal{D}}$ with the extracted values if this session is $\mathcal{G}^{\mathcal{O}'}$). Messages from $\mathcal{F}_{\mathrm{com}}^{\mathcal{O}}$ or $\mathcal{S}_{\mathcal{D}}$ to instances of $\mathcal{O}$ (which are "`ok`" messages) are suppressed. This way, the execution is consistent with the messages in the commit phase and distributed identically as if one of the protocols $\mathcal{G}^{\mathcal{O}'}$, $\rho^{\Pi_{\mathrm{S}},\mathcal{F}_{\mathrm{com}}^{\mathcal{O}}}$, or $\rho^{\mathcal{F}_{\mathrm{com}}^{\mathcal{O}}}$ was executed from the beginning.

Step 1. We show that

$$\mathrm{Exec}(\rho^{\Pi_{\mathrm{S}},\mathcal{F}_{\mathrm{com}}^{\mathcal{O}}}, \mathcal{Z}) \overset{\mathrm{c}}{\equiv} \mathrm{Exec}(\mathcal{G}^{\mathcal{O}'}, \mathcal{Z})$$

By a standard averaging argument, we can fix some random coins r for $\mathcal{Z}$. Thus we can assume henceforth that $\mathcal{Z}$ is deterministic.

Let q be an upper bound on the number of instances of $\rho^{\Pi_{\mathrm{S}},\mathcal{F}_{\mathrm{com}}^{\mathcal{O}}}$ that $\mathcal{Z}$ invokes. Consider the $2q+1$ hybrids $\mathsf{H}_{00}, \mathsf{H}_{01}, \mathsf{H}_{10}, \mathsf{H}_{11}, \mathsf{H}_{20}, \ldots, \mathsf{H}_{q0}$ which are constructed as follows:

Definition of hybrid H_{ij}: Execute the commit phase of each instance invoked by $\mathcal{Z}$ "without running the code of the parties" by invoking instances of $\mathcal{O}$. Follow the instruction of each instance of $\mathcal{O}$. Parties are only there as placeholders for the environment in the commit phases. Their instructions will be execute after the commit phase of the respective instance is over. Note that this can be done since the actions of the parties (i.e. choice of input values) in the commit phases have no effect on the view of the environment in these phases. Messages output from an instance of $\mathcal{O}$ are stored as well. After the commit phase of an instance is over do the following:

(See Fig. 4.8 on page 90 for a graphical depiction of the sequence of the hybrid games.)

1. If this is the kth instance in which the commit phase has ended and $k \leq i$ then invoke an instance of the dummy adverary simulator and the

functionality $\mathcal{G}$. Hand the dummy parties their respective inputs and the dummy adversary simulator the messages output by the instances of $\mathcal{O}$. Follow the instructions of the dummy adversary simulator and $\mathcal{G}$. Ignore messages of the dummy adversary simulator to the environment if these messages are coming from an instance of $\mathcal{F}_{\text{com}}$ in the commit phase (i. e. an "`ok`" message). In the unveil phase, messages from the dummy adversary simulator mimicking an interaction with $\mathcal{F}_{\text{com}}$ (which are messages of the form $(\texttt{Unveil}, b)$) are forwarded to the respective instance of $\mathcal{O}$ (with the same SID). Messages from the dummy adversary simulator not mimicking an interaction with an instance of $\mathcal{F}_{\text{com}}$ are output (without forwarding them to an $\mathcal{O}$-instance).

2. If $k = i + 1$ and $j = 0$ or $k > i + 1$ then run the protocol parties of $\rho^{\mathcal{F}_{\text{com}}}$ with their inputs and follow their instructions. For all subsessions where the sender is corrupted invoke instances of Π_{S} and "replay" the commit phase of Π_{S} using the same randomness for the receiver as the respective oracle and the messages the environment has sent. For all subsessions where the receiver or both or no party has been corrupted invoke instances of $\mathcal{F}_{\text{com}}$ and adjoin the respective oracle. Ignore "`ok`" messages from the instances of $\mathcal{F}_{\text{com}}$.

3. If $k = i + 1$ and $j = 1$ then run the parties of $\rho^{\mathcal{F}_{\text{com}}}$ with their inputs in the commit phase and follow their instructions. For all subsessions, invoke an instance of $\mathcal{F}_{\text{com}}$ and adjoin the respective oracle. Send the extracted commited values of the $\mathcal{O}$-instances in sender sessions to the respective $\mathcal{F}_{\text{com}}$-instance. Ignore "`ok`" messages from the instances of $\mathcal{F}_{\text{com}}$.

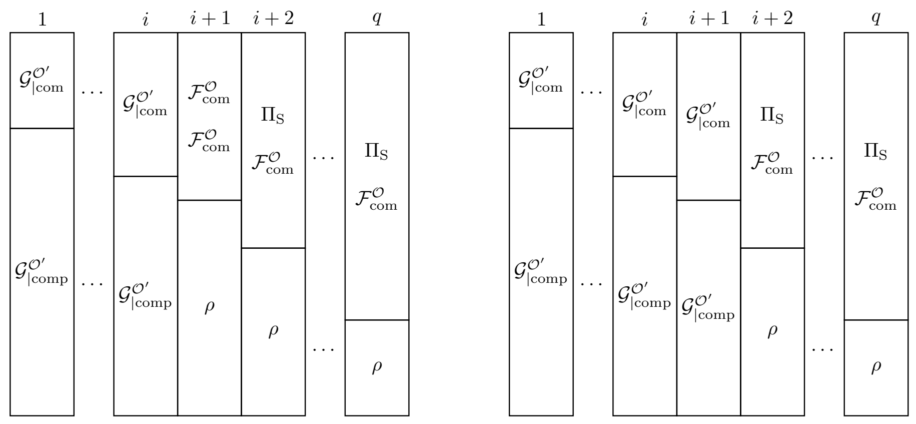
Observe that $\mathsf{H}_{00} = \text{Exec}(\rho^{\Pi_{\text{S}}, \mathcal{F}^{\mathcal{O}}_{\text{com}}}, \mathcal{Z})$ and $\mathsf{H}_{q0} = \text{Exec}(\mathcal{G}^{\mathcal{O}'}, \mathcal{Z})$.

Let P_{ij} denote the probability that $\mathcal{Z}$ outputs 1 in hybrid H_{ij}. Assume $|P_{00} - P_{q0}|$ is non-negligible. Then there exists an index i^* such that either $\left|P_{i^*1} - P_{(i^*+1)0}\right|$ or $|P_{i^*0} - P_{i^*1}|$ is also non-negligible.

Case 1: $\left|P_{i^*1} - P_{(i^*+1)0}\right|$ is non-negligible. In this case, these neighboring hybrids are equal except that in the $(i^* + 1)$th instance $\rho^{\mathcal{F}^{\mathcal{O}}_{\text{com}}}$ is replaced by $\mathcal{G}^{\mathcal{O}'}$.

We can now fix the coins that are used in the experiment up until the point where the $(i^* + 1)$th commit phase has ended, while maintaining $\mathcal{Z}$'s distinguishing advantage.

We can now construct an environment $\mathcal{Z}'$ that distinguishes $\rho^{\mathcal{F}_{\text{com}}}$ from $\mathcal{G}$. As a non-uniform advice, $\mathcal{Z}'$ receives a complete trace of all messages sent until this point, including all shares s_i and strings I that $\mathcal{Z}$ committed to until the point where the $(i^* + 1)$th commit phase has ended (note that since $\langle C, R\rangle$ is immediately committing, the first message in a receiver session uniquely determines the string I. Also note that the interaction with all $\mathcal{O}$-instances for the case of a corrupted sender mimicking a commit phase of Π is over until this point.) $\mathcal{Z}'$ internally simulates the execution experiment with $\mathcal{Z}$ using its advice. Messages to the $(i^* + 1)$th instance are sent to the challenge protocol. $\mathcal{Z}'$ may (tentatively) also invoke ideal receiver sessions in order to simulate ideal receiver sessions that are invoked after the point where the $(i^* + 1)$th commit phase has ended.


1
i
$i+1$
$i+2$
q
$\mathcal{G}^{\mathcal{O}'}_{|\mathrm{com}}$
$\mathcal{G}^{\mathcal{O}'}_{|\mathrm{comp}}$
$\mathcal{F}^{\mathcal{O}}_{\mathrm{com}}$
Π_{S}
ρ
$\dots$

(a) Hybrid H_{i1}: All instances up to and including i are ideal protocol instances, instance $i+1$ has a completely ideal commit phase but still a real compute phase, all instances from $i{+}2$ and above are $\rho^{\Pi_{\mathrm{S}},\mathcal{F}^{\mathcal{O}}_{\mathrm{com}}}$

(b) Hybrid $\mathsf{H}_{(i+1)0}$: All instances up to and including $i+1$ are ideal protocol instances, all instances from $i+2$ and above are $\rho^{\Pi_{\mathrm{S}},\mathcal{F}^{\mathcal{O}}_{\mathrm{com}}}$

Figure 4.8: The sequence of hybrids H_{ib}: The notation $\mathcal{G}^{\mathcal{O}'}_{|\mathrm{comp}}$ denotes the "compute phase" of a $\mathcal{G}$-instance and $\mathcal{G}^{\mathcal{O}'}_{|\mathrm{com}}$ denotes the "commit phase" of a $\mathcal{G}$-instance

Observe that the real execution corresponds to hybrid H_{i^*1} and the ideal execution to hybrid $\mathsf{H}_{(i^*+1)0}$. By construction, $\mathcal{Z}'$ distinguishes $\rho^{\mathcal{F}_{\mathrm{com}}}$ from $\mathcal{G}$. Since $\mathcal{F}^{\mathcal{O}}_{\mathrm{com}}$ is polynomially simulatable, $\mathcal{Z}'$ can be replaced by a polynomial time environment that also distinguishes $\rho^{\mathcal{F}_{\mathrm{com}}}$ from $\mathcal{G}$, using Theorem 4.15. We have thus reached a contradiction.

Case 2: $|P_{i^*0} - P_{i^*1}|$ is non-negligible. In this case, these neighboring hybrids are equal except that in the (i^*+1)th instance $\rho^{\Pi_{\mathrm{S}},\mathcal{F}^{\mathcal{O}}_{\mathrm{com}}}$ is replaced by $\rho^{\mathcal{F}^{\mathcal{O}}_{\mathrm{com}}}$. Since $\mathcal{Z}$ distinguishes these hybrids it holds that with non-negligible probability $\mathcal{Z}$ causes a *discrepancy* in hybrid H_{i^*1} as otherwise these hybrids would be statistically close. Let $\widetilde{\mathcal{Z}}$ be the environment that internally runs $\mathcal{Z}$ and outputs 1 as soon as a discrepancy occurs.[18] If no discrepancy occurs, $\widetilde{\mathcal{Z}}$ outputs 0. By

[18]Note that a subtlety arises here since the environment may not be able to learn the committed values and is therefore unable to detect discrepancies. To make the environment able to learn the committed values, we redefine the shielded oracle $\mathcal{O}$ for the case of a corrupted sender as follows: After the simulated unveil phase is over and accepting, the oracle first outputs the extracted committed value to the simulator. After receiving a notification message from the simulator, the oracle sends the `Unveil`-message to the functionality. Denote this modified oracle by $\widetilde{\mathcal{O}}$. Furthermore define $\widetilde{\Pi}$ to be identical to Π, except that the receiver sends the unveiled value to the sender after he has accepted this value. The sender then sends a notification message to the receiver who then outputs the unveiled value. It follows from the same arguments as in the proof of Theorem 4.26 that $\widetilde{\Pi} \geq_{\widetilde{\mathcal{F}^{\mathcal{O}}_{\mathrm{com}}}} \widetilde{\mathcal{F}^{\mathcal{O}}_{\mathrm{com}}}$ and that $\widetilde{\mathcal{F}^{\mathcal{O}}_{\mathrm{com}}}$ is polynomially simulatable. Using these modified versions in the above proof one obtains $\rho^{\widetilde{\Pi}} \geq_{\mathcal{G}^{\mathcal{O}'}} \mathcal{G}^{\mathcal{O}'}$. Since Π unconditionally emulates $\widetilde{\Pi}$ it holds that $\rho^{\Pi} \geq_{\mathcal{G}^{\mathcal{O}'}} \rho^{\widetilde{\Pi}}$, hence $\rho^{\Pi} \geq_{\mathcal{G}^{\mathcal{O}'}} \mathcal{G}^{\mathcal{O}'}$ by transitivity of $\mathcal{G}^{\mathcal{O}'}$-emulation.

construction, $\widetilde{\mathcal{Z}}$ outputs 1 with non-negligible probability in H_{i^*1}. We will now consider $i^* + 1$ new hybrids $\mathsf{h}_0, \ldots, \mathsf{h}_{i^*}$.

Definition of hybrid h_j: (See Fig. 4.9 on page 92 for a graphical depiction of the hybrids h_j.)

Execute the commit phase of each instance "without running the code of the parties" as described in the description of the hybrids H_{ij}. After the commit phase of an instance is over do the following (for a fixed $j \in \{0, \ldots, i^*\}$):

1. If this is the kth instance in which the commit phase has ended and $k \leq i^* - j$ then invoke an instance of the dummy adverary simulator and the functionality $\mathcal{G}$. (Marked as range (I) in Fig. 4.9.)

   Hand the dummy parties their respective inputs and the dummy adversary simulator the messages output by the instances of $\mathcal{O}$. Follow the instructions of the dummy adversary simulator and $\mathcal{G}$. Ignore messages of the dummy adversary simulator to the environment if these messages are coming from an instance of $\mathcal{F}_{\text{com}}$ in the commit phase (i. e. an "`ok`" message). In the unveil phase, messages from the dummy adversary simulator mimicking an interaction with $\mathcal{F}_{\text{com}}$ (which are messages of the form $(\texttt{Unveil}, b)$) are forwarded to the respective instance of $\mathcal{O}$. Messages from the dummy adversary simulator not mimicking an interaction with an instance of $\mathcal{F}_{\text{com}}$ are output (without forwarding them to an $\mathcal{O}$-instance).

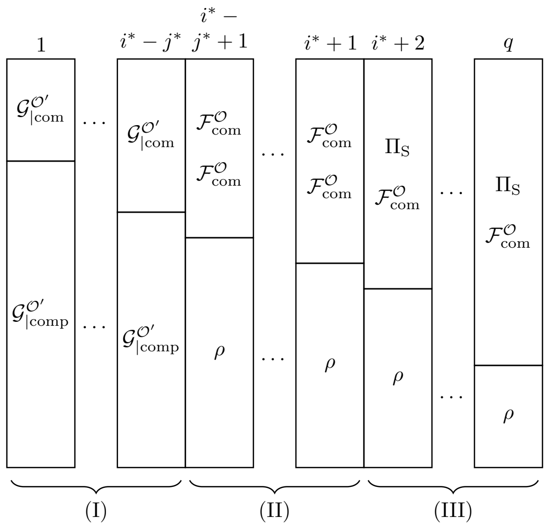
2. If $i^* - j + 1 \leq k \leq i^* + 1$ then run the protocol parties of $\rho^{\mathcal{F}_{\text{com}}}$ with their inputs in the commit phase and follow their instructions. (Marked as range (II) in Fig. 4.9.)

   For all subsessions, invoke an instance of $\mathcal{F}_{\text{com}}$ and adjoin the respective oracle. Send the extracted commited values of the $\mathcal{O}$-instances in sender sessions to the respective $\mathcal{F}_{\text{com}}$-instance. Ignore "`ok`" messages from the instances of $\mathcal{F}_{\text{com}}$.

3. If $k \geq i^* + 2$ then run the protocol parties of $\rho^{\mathcal{F}_{\text{com}}}$ with their inputs and follow their instructions. For all subsessions where the sender is corrupted invoke instances of Π_{S} and "replay" the commit phase of Π_{S} using the same randomness for the receiver as the respective oracle and the messages the environment has sent. For all subsessions where the receiver or both or no party has been corrupted invoke instances of $\mathcal{F}_{\text{com}}$ and adjoin the respective oracle. Ignore "`ok`" messages from the instances of $\mathcal{F}_{\text{com}}$.

Observe that $\mathsf{h}_0 = \mathsf{H}_{i^*1}$. Let j^* be the *largest index* such that $\widetilde{\mathcal{Z}}$ causes a discrepancy in hybrid h_{j^*} with non-negligible probability. j^* is well-defined, since there is an index for which this property holds (namely 0). Furthermore, $j^* \leq i^* - 1$. This follows from the following argument. Observe that the last hybrid h_{i^*} only contains instances of $\rho^{\mathcal{F}^{\mathcal{O}}_{\text{com}}}$. Since Π $\mathcal{F}^{\mathcal{O}}$-emulates IDEAL$(\mathcal{F}^{\mathcal{O}})$ and due to the composition theorem, Exec$(\rho^{\Pi}, \mathcal{Z})$ is indistinguishable from h_{i^*}. Since no discrepancy occurs in Exec$(\rho^{\Pi}, \mathcal{Z})$ it follows that a discrepancy can occur in h_{i^*} only with negligible probability.

By construction, $\widetilde{\mathcal{Z}}$ distinguishes the hybrids h_{j^*} and h_{j^*+1} (in the first hybrid $\widetilde{\mathcal{Z}}$ outputs 1 with non-negligible probability and in the second hybrid only with negligible probability).


1
$i^* - j^*$
$i^* - j^* + 1$
$i^* + 1$
$i^* + 2$
q
$\mathcal{G}^{\mathcal{O}'}_{|\mathrm{com}}$
$\mathcal{G}^{\mathcal{O}'}_{|\mathrm{comp}}$
$\mathcal{F}^{\mathcal{O}}_{\mathrm{com}}$
Π_{S}
ρ
(I)
(II)
(III)

(a) Hybrid h_{j^*}: The substitution of ρ by $\mathcal{G}$ has been reverted for instances $k \in \{i^* - j^* + 1, \ldots, i^* + 1\}$ (marked as range (II))

Figure 4.9: Hybrid h_j.

We will now modify these hybrids. For $k \in \{j^*, j^* + 1\}$ define the hybrid hyb_{k-j^*} to be identical to h_k except for the following: At the beginning, the experiment randomly selects one sender session. In all commit phases that end *after* the $(i^* - j^*)$th commit phase the real protocol Π_{S} is invoked instead of $\mathcal{F}^{\mathcal{O}_{\mathrm{S}}}$ in all sender sessions that have not been selected at the beginning. The one sender session that has been selected at the beginning remains ideal if and only if it is part of one of the commit phases $1, \ldots, i^* + 1$.

It holds that $\widetilde{\mathcal{Z}}$ also distinguishes hyb_0 from hyb_1. This is because $\widetilde{\mathcal{Z}}$ still causes a discrepancy in hyb_0 with non-negligible probability because with high probability ($1/poly$) the first sender session in which $\widetilde{\mathcal{Z}}$ causes a discrepancy is selected. Furthermore, $\widetilde{\mathcal{Z}}$ causes a discrepancy in hyb_1 with only negligible probability (otherwise $\widetilde{\mathcal{Z}}$ would have already caused a discrepancy with non-negligible probability in h_{j^*+1}).

We can now fix the coins that are used in the experiment up until the point where the $(i^* - j^*)$th commit phase has ended, while maintaining $\widetilde{\mathcal{Z}}$'s distinguishing advantage.

We can now construct an environment $\mathcal{Z}''$ that distinguishes $\rho^{\mathcal{F}_{\mathrm{com}}}$ from $\mathcal{G}$. As a non-uniform advice, $\mathcal{Z}''$ receives a complete trace of all messages sent until this point, including all shares s_i and index sets I that $\widetilde{\mathcal{Z}}$ commited to until the point where the $(i^* - j^*)$th commit phase has ended. $\mathcal{Z}''$ proceeds as follows: $\mathcal{Z}''$ internally simulates the execution experiment with $\widetilde{\mathcal{Z}}$ using its advice. Messages to the $(i^* - j^*)$th instance are sent to the challenge protocol. $\mathcal{Z}''$ can simulate the only instance of $\mathcal{F}^{\mathcal{O}_{\mathrm{S}}}$ that may occur in a commit phase with its parallel CCA oracle $\mathcal{O}_{\mathsf{pcca}}$ (note that if $\widetilde{\mathcal{Z}}$ has already started to commit to shares in this session then these shares are also part of $\mathcal{Z}''$'s advice.) $\mathcal{Z}''$ may (tentatively) also invoke ideal receiver sessions in order to simulate ideal receiver sessions that are invoked after the point where the $(i^* - j^*)$th commit phase has ended.

Observe that the real execution corresponds to hybrid hyb_1 and the ideal execution to hybrid hyb_0. By construction, $\mathcal{Z}''$ distinguishes $\rho^{\mathcal{F}_{\mathrm{com}}}$ from $\mathcal{G}$. With the same argument as in the proof of Theorem 4.26, step 1, case 2, one can replace all ideal receiver sessions that $\mathcal{Z}''$ invokes with instances of the real protocol (using the fact that $\langle C, R\rangle$ is parallel-CCA-secure). By construction, an environment $\mathcal{Z}''$ was found that can query the parallel CCA oracle $\mathcal{O}_{\mathsf{pcca}}$ and distinguish $\rho^{\mathcal{F}_{\mathrm{com}}}$ and $\mathcal{D}$ from $\mathcal{G}$ and $\mathcal{S}_{\mathcal{D}}$. We have thus reached a contradiction.

Step 2. We show that $\rho^{\Pi} \geq_{\mathcal{G}^{\mathcal{O}'}} \mathcal{G}^{\mathcal{O}'}$, completing the proof.

Let $\mathcal{Z}$ be a $\mathcal{G}^{\mathcal{O}'}$-augmented environments interacting with ρ^{Π} and the dummy adversary $\mathcal{D}$. By step 1, we can replace all instances of $\mathcal{G}^{\mathcal{O}'}$ with instances of $\rho^{\Pi_{\mathrm{S}}, \mathcal{F}^{\mathcal{O}}_{\mathrm{com}}}$. Since Π $\mathcal{F}^{\mathcal{O}}$-emulates IDEAL$(\mathcal{F}^{\mathcal{O}})$, it follows from the composition theorem that we can replace (the challenge protocol) ρ^{Π} also with $\rho^{\Pi_{\mathrm{S}}, \mathcal{F}^{\mathcal{O}}_{\mathrm{com}}}$. Again by step 1, we can replace all instances of $\rho^{\Pi_{\mathrm{S}}, \mathcal{F}^{\mathcal{O}}_{\mathrm{com}}}$ back with instances of $\mathcal{G}^{\mathcal{O}'}$. The theorem follows. □

If the following property holds for the commitment scheme $\langle C, R\rangle$, the premise $\rho^{\mathcal{F}_{\mathrm{com}}} \geq_{\mathcal{O}_{\mathsf{pcca}}\text{-}\mathsf{UC}} \mathcal{G}$ is automatically fulfilled.

Definition 4.35 (r-Non-Adaptive Robustness)**.** *Let $\langle C, R\rangle$ be a tag-based commitment scheme and $\mathcal{O}_{\mathsf{pcca}}$ its parallel CCA oracle. For $r \in \mathbb{N}$, we say that $\langle C, R\rangle$ is* r-non-adaptively-robust w. r. t. $\mathcal{O}_{\mathsf{pcca}}$ *if for every PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$, such that for every PPT r-round interactive Turing machine $\mathcal{B}$, the following two ensembles are computationally indistinguishable:*

- $\{\langle \mathcal{B}(y), \mathcal{A}^{\mathcal{O}_{\mathsf{pcca}}}(z)\rangle(1^n)\}_{n\in\mathbb{N}, y\in\{0,1\}^*, z\in\{0,1\}^*}$
- $\{\langle \mathcal{B}(y), \mathcal{S}(z)\rangle(1^n)\}_{n\in\mathbb{N}, y\in\{0,1\}^*, z\in\{0,1\}^*}$

The above definition is a relaxation of the (adaptive) robustness property put forward by [CLP10].

Corollary 4.36. *Let $\rho^{\mathcal{F}_{com}}$ be a constant-round CC protocol and $\mathcal{G}$ a functionality such that $\rho^{\mathcal{F}_{com}} \geq_{\mathsf{UC}} \mathcal{G}$. If the commitment scheme $\langle C, R\rangle$ in Π is additionally r-non-adaptively-robust for sufficiently large r, then there exists a shielded oracle $\mathcal{O}'$ such that*

$$\rho^{\Pi} \underset{\mathcal{G}^{\mathcal{O}'}}{\geq} \mathcal{G}^{\mathcal{O}'}$$

Up to now, we could instantiate $\langle C, R\rangle$ with a modified version of the scheme in [Goy+14] as described above of Corollary 4.28. This scheme can be easily made also r-non-adaptively-robust w. r. t. $\mathcal{O}_{\mathsf{pcca}}$ by adding "additional slots" using the technique in [LP09] (the scheme needs to have at least $r+1$ slots in order to be r-non-adaptively-robust).

In the following lemma, we show that every UC-secure protocol $\rho^{\mathcal{F}_{\mathrm{com}}}$ can be transformed into a UC-secure CC protocol.

Lemma 4.37 (CC-Compiler)**.** *Let $\rho^{\mathcal{F}_{com}}$ be a protocol in the $\mathcal{F}_{com}$-hybrid model. Then there exists a CC protocol* $\mathrm{Comp}(\rho)^{\mathcal{F}_{com}}$ *such that* $\mathrm{Comp}(\rho)^{\mathcal{F}_{com}} \geq_{\mathsf{UC}} \rho^{\mathcal{F}_{com}}$*. Furthermore, if $\rho^{\mathcal{F}_{com}}$ is constant-round then so is* $\mathrm{Comp}(\rho)^{\mathcal{F}_{com}}$*.*

Idea of proof. Replace each instance of $\mathcal{F}_{\text{com}}$ with a *randomized commitment* where the sender commits to a bit b by sending a random value a to $\mathcal{F}_{\text{com}}$ and $a \oplus b$ to the receiver. Since the number or rounds of $\rho^{\mathcal{F}_{\text{com}}}$ is polynomially bounded, the number of commitments of each party is also polynomially bounded. Put all randomized calls to $\mathcal{F}_{\text{com}}$ in a single commit phase. □

Let Π_r be the constant-round protocol as in Construction 1 where $\langle C, R\rangle$ is instantiated with the immediately committing, parallel CCA-secure and r-non-adaptively-robust modified version of [Goy+14] as described above. Furthermore, let Π_r^{BB} be the same as Π_r, except that [Goy+14] is instantiated with a verifiable perfectly binding homomorphic commitment scheme, thus making the construction fully black-box. Applying Corollary 4.36 and Lemma 4.37, one obtains the following:

Corollary 4.38. *Let $\rho^{\mathcal{F}_{com}}$ be a constant-round protocol and $\mathcal{G}$ a functionality such that $\rho^{\mathcal{F}_{com}} \geq_{\mathsf{UC}} \mathcal{G}$.*

(a) Assume the existence of one-way permutations. Then, for sufficiently large r, there exists a shielded oracle $\mathcal{O}'$ such that

$$\mathrm{Comp}(\rho)^{\Pi_r} \underset{\mathcal{G}^{\mathcal{O}'}}{\geq} \mathcal{G}^{\mathcal{O}'}$$

(b) Assume the existence of verifiable perfectly binding homomorphic commitment schemes. Then, for sufficiently large r, there exists a shielded oracle $\widetilde{\mathcal{O}}$ such that

$$\mathrm{Comp}(\rho)^{\Pi_r^{BB}} \underset{\mathcal{G}^{\widetilde{\mathcal{O}}}}{\geq} \mathcal{G}^{\widetilde{\mathcal{O}}}$$

4.5 Constant-Round (Black-Box) General MPC

We now apply Corollary 4.38 to obtain two constant-round general MPC protocols based on standard polynomial-time assumptions that are secure in our framework. By [Can+02; IPS08][19], there exists a UC-secure constant-round general MPC protocol (for realizing every standard well-formed functionality) in the $\mathcal{F}_{\text{com}}$-hybrid model based on enhanced trapdoor permutations . Plugging Π_r (for a sufficiently large r) into this protocol yields a (non-black-box) constant-round general MPC protocol in the plain model based on enhanced trapdoor permutations. Furthermore, by [HV15], there exists a UC-secure constant-round *black-box* general MPC protocol (for realizing every standard well-formed functionality) in the CRS-hybrid model based on IND-CPA-secure PKE schemes and constant-round semi-honest oblivious transfer.[20] This protocol can be transformed into a UC-secure, constant-round, black-box protocol in the $\mathcal{F}_{\text{com}}$-hybrid model based on IND-CPA-secure PKE schemes with oblivious public-key generation.[21,22] Plugging the black-box protocol Π_r^{BB} into this modified version of

[19]More specifically, it follows from [Can+02] that there exists a constant-round protocol UC-realizing $\mathcal{F}_{\mathsf{OT}}$ in the $\mathcal{F}_{\text{com}}$-hybrid model based on enhanced trapdoor permutations. Furthermore, [IPS08] provided a constant-round UC-secure general MPC protocol in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model based on pseudorandom generators (or equivalently one-way functions).

[20]The CRS is the public key of the underlying IND-CPA secure PKE scheme in the construction presented in [HV15].

[21]Cf. Definition 2.14 for a definition of PKE schemes with oblivious public key generation.

[22]Note that IND-CPA-secure PKE schemes with oblivious public-key generation imply two-round semi-honest oblivious transfer, cf. [Ger+00]).

[HV15] yields a fully black-box constant-round general MPC protocol in the plain model based on verifiable perfectly binding homomorphic commitment schemes and IND-CPA-secure PKE schemes with oblivious public-key generation.

Theorem 4.39 (Constant-round (Black-Box) General MPC in the Plain Model)**.**

(a) Assume the existence of enhanced trapdoor permutations. Then, for every standard well-formed[23] *functionality $\mathcal{F}$, there exists a constant-round protocol $\pi_{\mathcal{F}}$ in the plain model and a shielded oracle $\mathcal{O}$ such that*

$$\pi_{\mathcal{F}} \underset{\mathcal{F}^{\mathcal{O}}}{\geq} \mathcal{F}^{\mathcal{O}}$$

(b) Assume the existence of verifiable perfectly binding homomorphic commitment schemes and IND-CPA-secure PKE schemes with oblivious public-key generation. Then, for every standard well-formed functionality $\mathcal{F}$, there exists a constant-round protocol $\pi_{\mathcal{F}}^{BB}$ in the plain model and a shielded oracle $\mathcal{O}$ such that

$$\pi_{\mathcal{F}}^{BB} \underset{\mathcal{F}^{\mathcal{O}}}{\geq} \mathcal{F}^{\mathcal{O}}$$

Furthermore, $\pi_{\mathcal{F}}^{BB}$ uses the underlying commitment scheme and PKE scheme only in a black-box way.

[23]Cf. Section 2.4.2 for a definition of standard well-formed functionalities.

Chapter 5

A New Framework for Utilizing Simple Remotely Unhackable Hardware Modules

5.1 Introduction

In this chapter, we present a new framework that allows to adequately capture the advantages provided by remotely unhackable hardware modules. Before presenting our new framework, we first recall the adaptive UC framework and its shortcomings.

The Adaptive UC Framework. In the UC framework with adaptive corruptions ("adaptive UC framework") adversaries are allowed to corrupt parties throughout the entire UC execution experiment, enabling them to base their corruption strategy on their view so far (cf. Section 2.4.1). An adversary interacting with an ideal protocol ϕ is therefore able to learn and modify inputs and outputs of each party during the entire execution if he corrupts it (by sending `corrupt`-messages to the ideal functionality, cf. Section 2.4.1). As a consequence, an adversary interacting with a protocol π that UC-emulates ϕ may also have these possibilities.

Intuitively, the adaptive UC framework captures hackers breaking into computers during the execution of a protocol. Since direct physical attacks (i.e. tampering with hardware) are time consuming and therefore must typically be mounted before the start of the protocol execution, adaptive corruptions can be regarded as *remote attacks*, e.g., sending of computer viruses. The adaptive UC framework is meaningful in many settings. However, it ignores the realistic possibility of parties being (temporarily) isolated from the network and therefore not remotely hackable. For instance, a party could utilize *simple remotely unhackable hardware modules* such as *data diodes* to implement unidirectional channels or *air-gap switches* to disconnect itself from the network. An adversary may therefore not be able to corrupt a party via a remote attack during the entire protocol execution but only while the party is *online*, i.e. able to receive messages from the outside world. Furthermore, a party may have additional hardware modules at its disposal such as a simple encryption unit that only implements a specific public-key encryption scheme. Since such hardware modules with very limited functionality can be implemented securely as fixed-function circuits and formally verified for correctness, they can be assumed to be resilient against remote attacks. In particular, an adversary can only corrupt such hardware modules if he has direct physical access to them. Using such hardware modules in conjunction with air-gap switches and data diodes enables a party to, e.g., implement secure message transmission without risking to be remotely hacked while sending (sensitive) data.

It follows that the advantages (e.g. isolation properties) provided by simple remotely unhackable hardware modules cannot be adequately captured in the adaptive UC framework since this framework gives the adversary too much freedom over party corruption. In the following, we will propose a new framework that allows to capture these advantages.

New Framework: "Fortified UC" We present a new framework that provides a composable security notion and allows to adequately capture the advantages provided by remotely unhackable hardware modules. In our framework, called *Fortified UC*, one distinguishes between two kinds of corruptions, namely *physical attacks* and *online attacks*. Physical attacks model adversaries physically tampering with or replacing hardware. Online attacks model adversaries mounting remote attacks, such as sending computer viruses. Contrary to physical attacks, online attacks give the adversary control over a party only if the party is currently *online* and *not* assumed to be unhackable. A party's current online state is determined by the type and state of its *channels*, e.g., state of its air-gap switches. The hardware modules used in a protocol and their interconnections are part of what we call the *protocol architecture* in our framework.

Utilizing only very few and very simple remotely unhackable hardware modules, we construct general MPC protocols that protect against all online attacks

i) mounted *after* a party received its (first) input (unless *all* parties are corrupted), and

ii) mounted *before* a party received input if the attack comes from the "outside", i.e. from all channels except one at a party's input port.

More specifically, the parties in our protocols are disconnected from the outside world while waiting for input and can therefore not be corrupted through online attacks from the outside at that point. After receiving input, the parties authenticate, mask and share their secrets in such a way that mounting online attacks gives the adversary control over a party but not the ability to *learn the inputs or outputs (i.e. results of the MPC) of a party, nor to modify them* unless he gains control over *all* parties. This stands in contrast to the adaptive UC framework where an adversary may learn and modify the inputs and outputs of corrupted parties after they received input.[1] Although erasing parties seem

[1]If the parties are able to reliably erase local data, then an adversary in the adaptive UC framework may not learn all inputs of a corrupted party if he corrupts the party "too late". For instance, if the task to be realized is reactive (cf. Section 2.4.1) then the adversary may not learn past inputs. However, he may still be able to learn the current input and all future inputs.

necessary to achieve such a strong protection, we show that this assumption can be dropped using appropriate protocol architectures.

It is important to note that, unlike the hardware tokens proposed by [Kat07], the simple remotely unhackable hardware modules used in our constructions can be tampered if one has direct physical access to them. They can therefore not be passed to other (possibly malicious) parties but are only used and trusted by their owner. Hence, they are not sufficient to circumvent the impossibility results of [CF01; CKL03; Lin03; PR08; KL11]. Our general MPC protocols therefore rely on additional, well-established setup assumptions in combination with simple remotely unhackable hardware modules to achieve concurrent composability. Given all the aforementioned assumptions, our constructions provide the best possible protection against online attacks in a setting where parties cannot be protected while waiting for (their first) input.

This chapter is taken almost entirely from [Bro+18c], which is available as a technical report at the Cryptology ePrint Archive but has not been published elsewhere yet.

5.1.1 Contribution

We utilize realistic simple remotely unhackable hardware modules that, to the best of our knowledge, have so far not been used for secure multi-party computation. In the following, we list the results of this chapter.

New Composable Security Notion
We propose a new framework that, unlike previous frameworks, allows to adequately capture the advantages provided by remotely unhackable hardware modules. As with UC security, the security notion of our new framework is closed under protocol composition (Theorem 5.10). Furthermore, our new security notion is equivalent to adaptive UC security for protocols that do not use any remotely unhackable hardware modules (Theorem 5.9). As a consequence, UC-secure protocols can be used as building blocks in constructions in our new framework.

Protocols with Strong Security Guarantees Against Online Attacks
Using only very few and very simple remotely unhackable hardware modules, we construct general MPC protocols with very strong security guarantees against online attacks: An adversary is unable to learn or modify a party's inputs and outputs by mounting online attacks unless he gains control over a party via the input port *before* the party has received its (first) input (or gains control over *all* parties). We present a construction for non-reactive functionalities (Theorem 5.16) using only two simple remotely unhackable hardware modules (apart from air-gap switches and data diodes) per party and a protocol for reactive functionalities (Theorem 5.22) that uses only one additional simple remotely unhackable hardware module per party. Both constructions can be proven secure in our new framework for adversaries that gain control over all but one parties. We also present an augmentation of these constructions that allow simulation even in the case that all parties are under adversarial control (Theorem 5.21

and Theorem 5.23)[2]. For simplicity, we assume erasing parties in our protocols. However, we later show how this assumption can be dropped (cf. Section 5.6).

5.1.2 Related Work

The *adaptive corruption model*, first proposed in [Can+96], models adversaries that can corrupt parties at any point during the protocol execution. Numerous general MPC protocols achieving adaptive UC security have been constructed in the literature, see e.g. [Can+02; IPS08; IPS09; Lin09; Dac+13b; CGP15; GP15; HLP15; HV15; DKR15; CPV17]. Almost all of these works consider the *non-erasure* model, where honest parties are not assumed to be able to erase data reliably. A notable exception is [Lin09] who constructed a relatively efficient, constant-round general MPC protocol based on only enhanced trapdoor permutations in the *erasure model*, where honest parties are assumed to reliably erase data. Also, [HLP15] provided relatively simple and efficient protocols in the *partial erasure model*, where only *one* party is assumed to erase data reliably.

The *mobile adversary model* (also called *proactive model*), introduced by [OY91], models an adversary taking over a party— similar in spirit to our framework as "remote hacks"—and possibly *undoing* the corruption at a later point in time. There are only a few works on MPC in this setting. [OY91] provided a general MPC protocol in the stand-alone setting that is secure against mobile adversaries. [Bar+14] later provided a UC-secure construction. Both of these protocols assume that a majority of parties is honest throughout the protocol execution. Very recently, [Eld+18] constructed a general MPC protocol that is even secure against a dishonest majority.

Regarding remotely unhackable hardware modules in a broader context: [Gar+15] initiated a general study of *one-way secure computation* protocols over noisy channels in a setting where only one party speaks (e.g. via a data diode). [AMR14] made use of a remotely unhackable *equality check* hardware module to ensure the correct, UC-secure functioning of a *parallel firewall* setup in the case of one malicious firewall.

[Kat07] proposed *tamper-proof hardware tokens* as a plausible additional setup assumption for UC-secure protocols. Along this line of research, [Goy+10] showed strong feasibility results of what can be done with such tokens. Moreover, [Döt+13] showed that UC security is possible with a constant number of untrusted and resettable hardware tokens. Furthermore, [HPV17] constructed constant-round adaptively secure protocols which allow all parties to be corrupted.

Isolation is a general principle in IT security with lots of research on isolation through *virtualization*, see e.g. [Nem17]. Isolation in this way can be seen as a software analog of a trusted, remotely unhackable encryption module. Moreover, there is a wealth of literature on data exfiltration/side channel attacks to air-gaps including attacks based on acoustic, electromagnetic and thermal covert channels, cf. [ZGL18]. However, these works are not relevant to our work because they aim at protecting against outgoing communication from malicious internal

[2]Note that obtaining a protocol π that allows simulation even in the case that all parties are corrupted is important in the setting where instances of π are subroutines of an "outer" protocol ρ (not involving all parties of π). In this setting, it may happen that all parties of an instance of π become corrupted but not all parties of the outer protocol ρ. Security of ρ should still hold in this case.

parties, while we use data diodes/air-gap switches for the purpose of not being corrupted through messages coming from the outside. As an example, the Qubes OS provides strict separation between application domains, allowing to use an isolated GPG environment in a safe manner [Qub18].

5.2 The Fortified Universal Composability Framework

In this section, we present a new framework that allows to adequately capture the advantages provided by remotely unhackable hardware modules. Our framework builds on the UC framework (cf. Section 2.4).

5.2.1 Channels

Recall that in the UC framework, communication is modelled via `external write` instructions written on an ITI's outgoing message tape. A *control function* decides if the instruction is allowed. There are three different ways for ITIs to communicate: provide input, send a message, give subroutine output. This is modelled by `external write` instructions targeted at another ITI's input tape, incoming message tape or subroutine output tape, respectively (cf. paragraph "External Write Instructions" in Section 2.4).

In order to model protection mechanisms such as *air-gap switches* and *data diodes* (unidirectional communication) as well as the *online state* of a protocol party, we explicitly specify (possibly multiple) *channels* between ITMs that determine if communication between two ITMs is allowed and in which direction it is allowed. In particular, if there exists no channel between two ITMs then communication is not allowed between them. We assume that each channel has a unique identifier.

Channels can be between (sub-)parties of a protocol or between (sub)parties and ideal functionalities. In addition, channels can also be between a party and the environment or the adversary. Channels between a party and the environment model the allowed communication with calling parties from other protocols. Channels between a party and the adversary model possible communication to the "outside world".

Channels are modelled on top of the existing communication mechanism of the UC framework. Specifically, each protocol description must include a set of channels involving the protocol parties, which is part of the *protocol architecture* (cf. Section 5.2.3). The protocol architecture is given to the control function as an additional input. An `external write` instruction is allowed by the control function only if there exists a channel that allows the intended communication between the sending ITI and the receiving ITI. Otherwise, an `external write` instruction is silently dropped.

In our framework, we have three kinds of channels: *standard channels* that permanently allow bi-directional communication as well as two kinds of *enhanced channels*: *air-gap switches* and *data diodes*.

Enhanced Channels. In our framework, we want to capture possible security gains resulting from being isolated by forbidding certain communication and thus preventing corruption ("remote hacking") by the adversary. To this end, we introduce two kinds of enhanced channels:

1. *Data diodes* that allow communication in one direction only.
2. *Air-gap switches* that can be *connected* or *disconnected* by the party that operates them. Disconnected air-gap switches allow no data transmission at all. Connected ones allow bi-directional communication. Each air-gap switch has an *initial connection state* determined by the protocol architecture.

In order to model the current state of air-gap switches, we introduce a special *air-gap switch status tape* for each party containing the identifiers of each of its air-gap switches as well as its current state. A party can change the current state of each of its air-gap switches by writing on this tape. The control function gets the contents of each air-gap switch status tape as an additional input.

Communication between $\mathcal{A}$ and $\mathcal{Z}$ and $\mathcal{A}$ and Ideal Functionalities. As in the UC framework, the adversary and the environment may freely interact with each other. The same applies to the communication between the adversary and ideal functionalities. Formally, we always assume standard channels between these ITIs which are given to the control function in addition to the protocol architecture. Communication between these ITMs is therefore independent of the given protocol architecture.

Terminology. Let μ and μ' be two ITIs. We say that "μ is connected to μ'" if there is a channel between μ and μ'. If there is a data diode between μ and μ' in the direction of μ' then we say that "μ is connected to μ' via data diode". If there is an air-gap switch operated by μ to μ' then we say that "μ is connected to μ' via air-gap switch". Likewise, we say that "μ is connected to μ' via a standard channel" if there is a standard channel between μ and μ'. If there is a channel C between μ and the adversary we say that "μ is connected to the adversary via C". Likewise, if there is a channel C between μ and the environment we say that "μ is connected to the environment via C". Furthermore, we say that "μ can send messages (or provide input or give output) to μ' via C" or that "μ' can receive messages (or input or output) from μ via C" if C is a channel between μ and μ' that allows the respective `external write` instruction.

Conventions for Graphical Depiction of Architectures. Main parties are represented by boxes with rounded corners, sub-parties and ideal functionalities by cornered ones. Boxes with bold lines and grey background denote that the sub-party is unhackable. Standard channels are denoted by lines, data diodes by —▷|— and air-gap switches by —⁄—(initially disconnected) and —⁄̵— (initially connected). Dashed lines denote standard channels to other parties that are not shown.

5.2.2 Online State

Online State of Channels to the Environment. The environment $\mathcal{Z}$ may, upon each activation, mark each channel that exists between $\mathcal{Z}$ and a protocol party either online or offline. For this, we introduce a special *channel marking tape* containing the identifiers of each channel to $\mathcal{Z}$ and the current markings. $\mathcal{Z}$ can change the current markings by writing on this tape. The control function gets the contents of this tape as an additional input. As the environment embodies other, concurrently executed protocols, this mechanism reflects the online state of the calling parties being implicitly incorporated in the environment. In addition, for each channel between $\mathcal{Z}$ and a party, $\mathcal{Z}$ is informed upon each activation about whether it can *receive output* via that channel[3].

Online State of Protocol Parties. A (sub-)party P of protocol π is *online via C* if C is a channel such that one of the following holds:

1. P can receive messages from the adversary via C
2. P can receive output from an ideal functionality $\mathcal{F}$ via C
3. P can receive output/input via C from a sub-party/calling party M and M is online via C' and C' is a channel between M and an ITM $\mu \neq P$.
4. P can receive input from the environment $\mathcal{Z}$ via C and $\mathcal{Z}$ has marked the channel C online

If none of the above holds, P is *offline via C*. If there exists no channel such that P is online via that channel, we say that P is *offline*. If P is online via some channel, we say that *P is online*.

Intuitively, (1) models a party who is able to receive messages from the "outside world" and is therefore online. (2) models a party who is able to receive messages from a trusted third party $\mathcal{F}$ that "lives" somewhere in the outside world.[4] For instance, $\mathcal{F}$ could be a public bulletin board, a common reference string, or a trusted party evaluating a specific function on the parties' private inputs. (3) models a party being *transitively* online via connections to a sub-party or calling party who is online. (4) models a party being transitively online via connections to a calling party from another protocol.

Note that each party has an *initial online state* prior to invocation depending on the protocol architecture (in particular, the initial connection states of air-gap switches) and how the environment initially marked the respective channels.

Status Report to the Adversary. Each time the adversary is activated, he gets informed via which channels each party is online. This is called the *status*. As will be described in Section 5.2.3, the adversary is able to gain control over "hackable" parties during the protocol execution when these parties are online. Giving the status to the adversary facilitates corruption as he does not have to examine which parties are online. Also, the status will play an important role in the proof of the composition theorem (Theorem 5.10).

[3]Jumping ahead, these two abilities of the environment will be very important in the proof of the composition theorem of our new framework (cf. Theorem 5.10).

[4]Note that it may be necessary to disable a party being online via a channel to specific functionalities such as signature cards in order to adequately model them. This can be done by, e.g., allowing functionalities to mark their channels to parties offline or online (like the environment). For simplicity, we do not consider this mechanism in this work.

Example 1. Consider the protocol architecture depicted in Fig. 5.1.

Let $\mathcal{Z}$ be an environment that permanently marks the channel to P_1 online and the channel to P_2 offline. P_1 disconnects its air-gap switch to $\mathcal{Z}$ as soon as it has received input. Later, P_1 connects its air-gap switch to the adversary $\mathcal{A}$ at a specific point, say, after having erased its input.

Q_1 is always online (being connected to the ideal functionality $\mathcal{F}$ via a standard channel, cf. (2) on Page 103). The same holds for P_2 (being connected to the adversary $\mathcal{A}$ via standard channel, cf. (1) on Page 103). Therefore, Q_2 is also always online (being connected to P_2 via a standard channel, cf. (3) on Page 103). P_1 is online before receiving its input (being connected to the environment $\mathcal{Z}$ via a connected air-gap switch and $\mathcal{Z}$ has marked the channel to P_1 online, cf. (4) on Page 103), offline immediately afterwards, and online again after having erased its input (having connected its air-gap switch to the adversary $\mathcal{A}$, cf. (1) on Page 103). M's online state is the same as P_1's (being connected to P_1 via a standard channel, cf. (3) on Page 103).

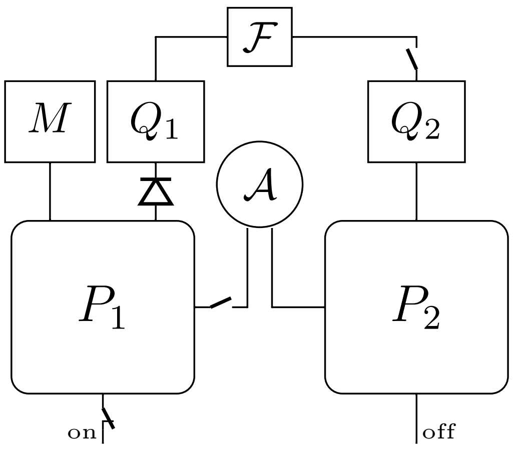


Figure 5.1: Protocol Architecture for Example 1

5.2.3 Corruption Model

We distinguish between two kinds of corruption: *physical attacks* and *online attacks*. Physical attacks model an adversary physically tampering with or replacing a party's hardware. Online attacks model remote hacks (e.g. sending a computer virus).

Since, in practice, physical attacks are time consuming and therefore typically must be mounted before the protocol execution begins, the adversary is allowed to carry out this type of corruption only *prior* to the start of the protocol execution in our model.[5] Online attacks, on the other hand, can be carried out throughout the protocol execution. Unlike physical attacks, online attacks only take effect if the targeted party is online and assumed to be *hackable*. In the following, we describe our new corruption model in more detail.[6]

[5]We note that our general MPC protocols for the case of up to $N-1$ corrupted parties (cf. Sections 5.4 and 5.5) can be shown to be secure even if physical attacks are allowed throughout the protocol execution.

[6]Note that the following describes the behavior of protocol parties in the real model upon

In our framework, parties can be either *hackable* or *unhackable.* The protocol architecture specifies which parties are hackable or unhackable (cf. Page 107).

Let $\mathcal{P}$ be the set of main parties of a protocol π. At the first activation,[7] the adversary $\mathcal{A}$ may only send a `physical-attack` instruction that enables him to gain control over parties regardless of the protocol architecture. Formally, $\mathcal{A}$ writes $(\texttt{physical-attack}, \mathcal{M})$, where $\mathcal{M} \subseteq \mathcal{P}$, on his outgoing message tape. Each $P \in \mathcal{M}$ and all of their sub-parties are then connected to the adversary via a standard channel and all air-gap switches controlled by and data diodes coming from these parties are replaced with standard channels. From then on, $\mathcal{A}$ has full control over all $P \in \mathcal{M}$ and *all* of their sub-parties (including the unhackable ones).[8]

From the second activation on, $\mathcal{A}$ may not send a `physical-attack` instruction anymore. $\mathcal{A}$ may send `online-attack` instructions that enable $\mathcal{A}$ to gain control over *hackable* parties when they are online. Formally, if $\mathcal{A}$ writes $(\texttt{online-attack}, P)$ on his outgoing message tape and P is a (sub-)party of π that is online and hackable, then a standard channel between P and $\mathcal{A}$ is created and all air-gap switches controlled by P are connected. P then sends its entire local state to $\mathcal{A}$. From then on, $\mathcal{A}$ has full control over P. If P is *unhackable*, then this instruction is ignored.

If $\mathcal{A}$ has gained control over a (sub-)party P through one of the above instructions, we say that P is "corrupted".

Finally, if a (sub-)party P is corrupted, then each ideal functionality which is connected to P is informed about P being corrupted through a special message $(\texttt{corrupt}, P)$ that is written on its incoming message tape. Also, each main party immediately informs the environment after being corrupted.[9]

Example 2. Consider the protocol architecture in Fig. 5.2 on Page 106.

Let $\mathcal{Z}$ be an environment that permanently marks its channel to P_1 online and to P_2 $\mathsf{offline}$. On receiving input, P_1 disconnects its air-gap switch to $\mathcal{Z}$. P_2 connects its air-gap switches to the adversary and Q_2 upon receiving input.

At his first activation, the adversary $\mathcal{A}$ may write $(\texttt{pyhsical-attack}, \mathcal{M})$, $\mathcal{M} \subseteq \{P_1, P_2\}$. If, e.g., $\mathcal{M} = \{P_1\}$, then $\mathcal{A}$ gains control over P_1 and M as well as (the unhackable) party Q_1. Alternatively, from the second activation on, $\mathcal{A}$ may gain control over P_1 before P_1 has received its input by writing $(\texttt{online-attack}, P_1)$ (because P_1 is online via the channel to $\mathcal{Z}$ at this point). $\mathcal{A}$ may also choose to "skip" P_1 by writing $(\texttt{online-attack}, M)$ but not $(\texttt{online-attack}, P_1)$. This way, $\mathcal{A}$ can still gain control over P_1 after P_1 has received its input because P_1 is online via the channel to M (because a standard

corruption. As in the UC framework, party corruption in ideal protocols is handled solely by the ideal functionality.

[7]As in the UC framework, the first ITI to be invoked by the environment in our framework is the adversary (cf. Definition 5.3 for the Fortified UC execution experiment).

[8]Formally, these parties forward all future inputs and subroutine outputs to $\mathcal{A}$ and $\mathcal{A}$ can send these parties any instruction by writing `external write` instructions targeted at their incoming message tape. Note that since standard channels have been created between these parties and $\mathcal{A}$, these `external write` instructions are always allowed by the control function.

[9]Note that, to ensure that the above instructions can be fully carried out, the environment $\mathcal{Z}$ is not allowed to activate any other ITI until $\mathcal{Z}$ is explicitly informed (by the control function) that this instruction has been fully carried out (in particular, all functionalities connected to parties corrupted through that instruction have been (iteratively) informed that these parties are corrupted and all corrupted main parties have informed $\mathcal{Z}$ that they are corrupted).

channel between M and $\mathcal{A}$ has been created). Moreover, $\mathcal{A}$ cannot gain control over P_2 through an `online-attack` instruction before P_2 has received its input (because P_2 is offline up to this point).

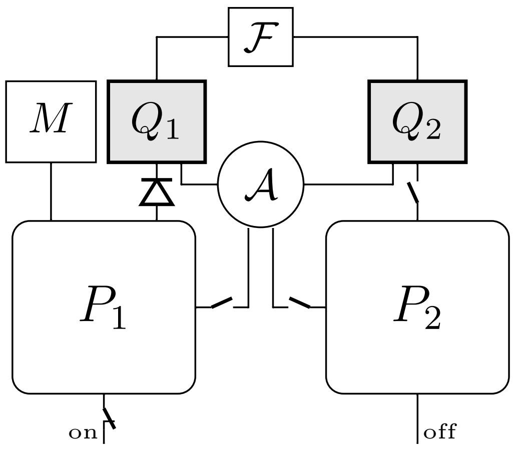


Figure 5.2: Protocol Architecture for Example 2

In addition to our new corruption model, we also introduce "`taint` instructions" and define when parties are "combined" (i.e. can only be corrupted together). Note that since these mechanisms play only a minor role in the rest of this chapter the following two paragraphs may be skipped on first reading.

"Tainting" Unhackable Parties. Consider an unhackable party E that is connected to a hackable party M via air-gap switch and to the adversary via air-gap switch. E's air-gap switch to M is connected only if E's air-gap switch to the adversary is disconnected. M is only connected to E. Therefore, $\mathcal{A}$ cannot gain control over M through an `online-attack` instruction since M is always offline. However, it should be intuitively possible for $\mathcal{A}$ to gain control over M since otherwise E would act as a "perfect firewall" for M.

To this end, $\mathcal{A}$ may send `taint` instructions in our model. Formally, if $\mathcal{A}$ writes $(\texttt{taint}, P)$ on his outgoing message tape and P is a (sub-)party of π that is online and unhackable, then a standard channel between P and $\mathcal{A}$ is created.

In the above example, writing a `taint` instruction when E's air-gap switch to $\mathcal{A}$ is connected enables $\mathcal{A}$ to gain control over M since M is then online via the air-gap switch to E when that air-gap switch is connected.

Combination of Parties. In the UC framework, parties may be combined by giving them the same PID or the same value in a component of the PID ("PID-wise corruption"). Intuitively, combined parties are processes running on the same physical machine. In our framework, two parties are *combined* if i) their PIDs are equal in the first component[10] and ii) they are connected via standard channels only and iii) they are both either hackable or unhackable. If two parties P, P' are combined then any $(\texttt{online-attack}, \mu)$ or $(\texttt{taint}, \mu)$ instruction such that $\mu \in \{P, P'\}$ affects both parties.

[10]Here we assume that PIDs have the form $(\texttt{pid}_1||\dots||\texttt{pid}_l)$, where $l \geq 1$, $\texttt{pid}_i \in \{0,1\}^*$.

We will later implicitly combine dummy parties with their respective calling party in the constructions presented in this chapter.

Remark 5.1. *Note that our corruption model gives the adversary lots of freedom. In particular, the adversary is still able to freely control a party he has corrupted via an* `online-attack` *instruction even if that party is offline via all channels specified by the protocol architecture (as is the case for the party M after P_1 has received its input in Example 2). This is possible because we grant the adversary standard channels to parties corrupted via* `online-attack` *instructions. Intuitively, this models the ability of a hacked device to communicate with the outside world via side-channels. Allowing the adversary to corrupt a party when it is online only via channels to a* tainted *party can also be seen as exploiting side channels. Also, the adversary always knows which parties are online and can gain control over a party also when that party is transitively online via some channel to another party. Our corruption model therefore captures the vulnerabilities implied by being online in a very pessimistic way. This has the advantage of making the security notion of our framework both strong and simple at the same time.*

Protocol Architecture. The *protocol architecture* of a protocol π consist of the following: i) a set of channels involving parties of π and ii) a specification of the initial connection state of each air-gap switch in that set and iii) for each party in π a specification of whether that party is hackable or unhackable.

5.2.4 Interface Modules and Fortified Functionalities

Recall our security goal stated in the introduction to this chapter: The adversary should be unable to learn or modify a party's inputs and outputs (i.e. results of the MPC) via `online-attack` instructions i) mounted *after* a party received its (first) input (unless *all* parties are corrupted) and ii) mounted before a party received input if the attack "comes from the outside". To model this goal, we introduce *interface modules*, an appropriate *ideal-model protocol architecture* and *fortified functionalities*.

Interface Modules. In order to achieve the above-mentioned level of security, a party's result of the MPC must remain unmodified and hidden from the adversary even if the party is corrupted via an `online-attack` instruction *after* receiving input. This is not possible if a party learns its result and outputs it itself since the adversary would then learn this result if he corrupts the party and could then also instruct the party to output a value that does not equal its result. Furthermore, for reactive tasks, a party corrupted via an `online-attack` instructio after receiving input n must also not be able to learn or modify its input(s) for the rounds ≥ 2.

Deviating from the UC framework, we therefore allow the main parties to invoke special sub-parties called *interface modules* that are connected to their main party as well as to the environment via channels specified by the protocol architecture. These interface modules may thus give subroutine output to or receive input from the environment subject to the protocol architecture.

Intuitively, interface modules model simple hardware modules connected to, e.g., a PC. During the protocol execution, a user does not trust his PC since it may have been remotely hacked (in particular, the output of his PC may have been altered by a hacker). Instead, he only trusts the unhackable interface modules and, in particular, the outputs given by them (e.g. via a display).

In our constructions, interface modules will be unhackable sub-parties with very limited functionality (except for the interface modules S_i introduced in Section 5.6 which will be hackable). We will assume an interface module called *output interface module* (OIM) that is used for ensuring that a party's result of the MPC remains unmodified and hidden from the adversary even in the case that the party is corrupted after receiving input. More specifically, a party's result(s) will only be learned by its OIM, which outputs these result(s) instead of the party. For reactive tasks, we will also assume an *input interface module* (IIM) for ensuring that a party's input(s) for the rounds ≥ 2 remain secret and unmodified even in the case that the party is corrupted after receiving (its first) input. Note that in the ideal execution, the ideal functionality may also interact with dummy parties corresponding to interface modules (see Definition 5.2).

Ideal Protocols. In ideal protocols, each dummy party is connected to the environment and to the ideal functionality $\mathcal{F}$ via channels specified by the ideal protocol's architecture. Recall that, as described in Section 5.2.3, $\mathcal{F}$ is informed through a special message $(\texttt{corrupt}, P)$, which is written on its incoming message tape, when a party P connected to $\mathcal{F}$ is corrupted.[11]

In this chapter, we will consider the ideal protocols $\texttt{SC}(\mathcal{F})$ and $\texttt{AG}(\mathcal{F})$, which are defined as follows:

$\texttt{SC}(\mathcal{F})$ is the ideal protocol where the dummy parties are connected to $\mathcal{F}$ and the environment via standard channels.

For a *non-reactive*[12] functionality $\mathcal{F}$, $\texttt{AG}(\mathcal{F})$ is the ideal protocol where N *hackable* "dummy main parties" $P_1, \ldots, P_N$ are connected to $\mathcal{F}$ via initially *disconnected* air-gap switches and to the environment via initially *connected* air-gap switches, and additionally N *unhackable* "dummy output interface modules" $\text{OIM}_1, \ldots, \text{OIM}_N$ are connected to $\mathcal{F}$ and the environment via standard channels (for a graphical depiction of $\texttt{AG}(\mathcal{F})$, see Fig. 5.3). Upon input v, each party P_i disconnects its air-gap switch to the environment, connects its air-gap switch to $\mathcal{F}$, and passes v to $\mathcal{F}$. Each P_i connects its air-gap switch to the environment again upon receiving a special message `open` from $\mathcal{F}$. Furthermore, if $\mathcal{F}$ is *reactive*, $\texttt{AG}(\mathcal{F})$ additionally contains N *unhackable* "dummy input interface modules" $\text{IIM}_1, \ldots, \text{IIM}_N$ which are connected to $\mathcal{F}$ via standard channels and to the environment via initially *disconnected* air-gap switches. Each IIM_i connects its air-gap switch to the environment upon receiving `open` from $\mathcal{F}$.

Note that since the air-gap switch between a party P_i and $\mathcal{F}$ is disconnected before P_i has received input, the parties P_i in $\texttt{AG}(\mathcal{F})$ cannot be corrupted by an `online-attack` instruction "coming from the outside" prior to receiving input. More specifically, each P_i can only be corrupted by an `online-attack` instruction prior to receiving input if it is online via its channel to the environment (which is the case if the environment has marked this channel `online`).

Note that in the following we will also refer to OIM_i (and IIM_i) as the "dummy OIM (resp. IIM) of P_i"

[11]Note that the adversary is not allowed to write `external write` instructions containing the special message (corrupt, P) in order to prevent him from bypassing the corruption rules (e.g. by sending $(\texttt{corrupt}, P)$ to $\mathcal{F}$ during the protocol execution while party P is offline).

[12]For a definition of reactive resp. non-reactive functionalities, see Section 2.4.1.

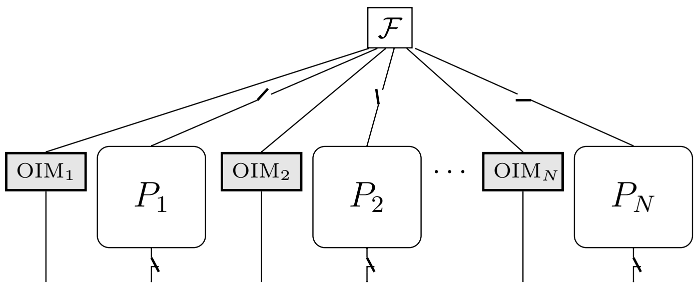


Figure 5.3: Protocol Architecture of the Ideal Protocol $\texttt{AG}(\mathcal{F})$ (Non-Reactive Case).

Fortified Functionalities. In contrast to functionalities in the adaptive UC framework, *fortified functionalities* do not pass the inputs and outputs of a party P_i corrupted *after* receiving input to the adversary $\mathcal{A}$ and also do not allow him to modify P_i's input and the *output to P_i's dummy* OIM, unless all parties P_j $(j = 1, \ldots, N)$ are corrupted. $\mathcal{A}$ can only block an output or instruct the functionality to pass either the output computed by the functionality or an error symbol $\perp$ to P_i's dummy OIM. If all parties are corrupted, $\mathcal{A}$ learns all inputs and outputs and may modify them arbitrarily (including the outputs to the dummy OIMs).

Definition 5.2 (Fortified Functionality)**.** *Let $\mathcal{G}$ be a* non-reactive *standard*[13] *ideal functionality interacting with N parties $P_1, \ldots, P_N$ and $\mathcal{A}$. Define the* fortified functionality $[\mathcal{G}]$ *of $\mathcal{G}$ interacting with $P_1, \ldots, P_N$, $\mathcal{A}$ and additionally N "dummy output interface modules* (OIMs)*"* OIM$_1, \ldots,$ OIM$_N$ *as follows:*

- $[\mathcal{G}]$ *internally runs an instance of $\mathcal{G}$.*
- $[\mathcal{G}]$ *initializes a counter $c = 0$.*
- *Upon receiving input from a party P_i, $[\mathcal{G}]$ forwards that input to $\mathcal{G}$.*
- *Each time $\mathcal{G}$ sends a notification to $\mathcal{A}$ upon receiving input from an (honest) party, $[\mathcal{G}]$ forwards that notification to $\mathcal{A}$.*
- *When $\mathcal{G}$ gives a (public or private) delayed output, $[\mathcal{G}]$ gives the same (public or private) delayed output with the only difference that, upon confirmation by $\mathcal{A}$, $[\mathcal{G}]$ forwards the output to the dummy* OIM *of the party for which $\mathcal{G}$ intended this output.*
- *Upon receiving* $(\texttt{corrupt}, P_i)$*, $[\mathcal{G}]$ does the following:*
  - *If $[\mathcal{G}]$ has* not yet *received input from P_i, $[\mathcal{G}]$ increments c, marks P_i as* corrupted before input *and forwards* $(\texttt{corrupt}, P_i)$ *to $\mathcal{G}$.*
  - *If $[\mathcal{G}]$ has already received input from P_i, $[\mathcal{G}]$ increments c, marks P_i as* corrupted after input *and forwards* $(\texttt{corrupt}, P_i)$ *to $\mathcal{G}$.*

[13] Cf. Section 2.4.2 for a definition of standard functionalities.

- *If $\mathcal{G}$ outputs "corrupted" to P_i upon receiving* $(\texttt{corrupt}, P_i)$*, $[\mathcal{G}]$ forwards this to P_i.*[14]
- *Handling Parties P_i marked as* corrupted before input*:*
  - *If $\mathcal{G}$ sends the input of P_i to $\mathcal{A}$, $[\mathcal{G}]$ forwards that input to $\mathcal{A}$. Furthermore, if $\mathcal{A}$ sends a modified input value for P_i, $[\mathcal{G}]$ forwards that value to $\mathcal{G}$.*
  - *If $\mathcal{G}$ sends an output intended for P_i to $\mathcal{A}$, $[\mathcal{G}]$ sends that output to $\mathcal{A}$. $\mathcal{A}$ may instruct $[\mathcal{G}]$ to pass any output of his choice to OIM_i.*
- *Handling Parties P_i marked as* corrupted after input*:*
  - *If $c < N$ and $\mathcal{G}$ sends the input of P_i to $\mathcal{A}$ upon receiving* $(\texttt{corrupt}, P_i)$ *(after having output "corrupted" to P_i), ignore this message. Furthermore, if $\mathcal{A}$ sends a modified input value for P_i, ignore this value.*
  - *If $c < N$ and $\mathcal{G}$ sends the output intended for P_i to $\mathcal{A}$, $[\mathcal{G}]$ first notifies $\mathcal{A}$ that OIM_i is about to receive output. $\mathcal{A}$ may then instruct $[\mathcal{G}]$ to pass that output or $\perp$ to OIM_i.*
- *If $c = N$, send all inputs and outputs to $\mathcal{A}$. In addition, $\mathcal{A}$ may determine the outputs of all dummy OIMs in this case.*
- *All other messages between $\mathcal{A}$ and $\mathcal{G}$ are forwarded.*
- *If $\mathcal{A}$ sends* $(\mathsf{output}, \tilde{y}, P_i)$*, $[\mathcal{G}]$ outputs $\tilde{y}$ to P_i if $[\mathcal{G}]$ has marked P_i.*[15]

Reactive Case. If $\mathcal{G}$ is *reactive*, then $[\mathcal{G}]$ is defined as above except that $[\mathcal{G}]$ additionally interacts with N "dummy input interface modules"$\mathrm{IIM}_1, \ldots, \mathrm{IIM}_N$ as follows: Upon receiving input from an honest party P_i, $[\mathcal{G}]$ forwards that input to $\mathcal{G}$ and sends `open` to the dummy IIM of P_i. If P_i is marked, $[\mathcal{G}]$ forwards all inputs provided by a party P_i for rounds $u \geq 2$ to the adversary $\mathcal{A}$. Furthermore, upon receiving an input provided by the dummy IIM of a party P_i who is marked as corrupted before input, $[\mathcal{G}]$ forwards this input to $\mathcal{A}$ who may then modify it. However, upon receiving an input provided by the dummy IIM of a party P_i who is *not* marked as corrupted before input, $[\mathcal{G}]$ does not forward this input to $\mathcal{A}$ (and does not allow $\mathcal{A}$ to modify it).

By construction, $\texttt{AG}([\mathcal{G}])$ captures our desired security goal: i) $[\mathcal{G}]$ ensures that corrupting a party P_i via an `online-attack` instruction *after* it has received its (first) input does not enable the adversary to learn or modify P_i's input(s) and result(s) of the MPC (i.e. outputs of P_i's dummy OIM), unless *all* parties P_j $(j = 1, \ldots, N)$ are corrupted, and ii) the initially disconnected air-gap switches between the parties P_i and $[\mathcal{G}]$ ensure that the adversary cannot corrupt a

[14]Note that this output is always allowed by the control function. In particular, it is allowed when a party is corrupted before receiving input because the respective air-gap switch, which is disconnected at that point, is replaced by a standard channel if the party is corrupted via a `physical-attack` instruction or immediately connected if a party is corrupted via an `online-attack` instruction.

[15]Note that the adversary is able to determine what a corrupted party outputs. However, he cannot modify the output of the (unhackable) dummy OIM of a party corrupted after receiving input (unless $c = N$).

party P_i via an `online-attack` instruction "coming from the outside" before P_i has received input, i.e. each P_i can only be corrupted via an `online-attack` instruction at that point if it is online via its channel to the environment.

5.2.5 Notify Transport Mechanism and Activation Instructions

In the UC framework, the adversary is not activated when a party provides input or receives subroutine output from a sub-party and is therefore not able to adaptively corrupt it during this communication. In our setting of hacking adversaries, this is undesirable because it does not capture the possibility of parties being remotely hacked when they are online during such communication.

As a motivating example, consider a hackable party P that is connected to the environment and the adversary $\mathcal{A}$ via standard channels. Furthermore, P is also connected to an unhackable sub-party P' via a standard channel. Upon receiving input, P provides an input containing secret data (e.g. shares of its input) to P'. P' then outputs a notification message to P who immediately *erases* all secret data after being activated again. As this message delivery is *immediate*, i.e. $\mathcal{A}$ is not activated during the communication between P and P', he is unable to corrupt P before P has erased its secret data and sent it to an unhackable sub-party, even though P has been *online all the time*.

To address this problem, we introduce a *notify transport mechanism* that activates $\mathcal{A}$ (under certain conditions) upon immediate message delivery.

Notify Transport Mechanism. Let μ, μ' by two ITIs such either

- μ and μ' are protocol (sub-)parties that are not combined, or
- μ is a protocol (sub-)party and μ' is an ideal functionality that is not a fortified functionality, or vice versa.

If μ sends an `external write` instruction addressed to μ' and the control function allows this instruction, then the adversary $\mathcal{A}$ is activated with a *notify transport message* (`notify transport`, μ''s ID). Upon activation with a notify transport message, $\mathcal{A}$ may forward the notify transport message to the environment $\mathcal{Z}$ or execute an `online-attack` or `taint` instruction (or do nothing, i.e. only carry out local computations and then go into idle mode). Upon activation with a notify transport message, $\mathcal{Z}$ may only activate $\mathcal{A}$ again. Upon activation by $\mathcal{Z}$ after having been activated with a notify transport message in the previous activation, $\mathcal{A}$ may only carry out an `online-attack` or `taint` instruction (or do nothing). Afterwards, the `external write` instruction is carried out.

Note that, upon receiving a notify transport message, $\mathcal{A}$ is not allowed to block the message sent by μ or activate another party.

Why Exclude Fortified Functionalities? The notify transport mechanism does not apply to the communication between the dummy parties and $[\mathcal{G}]$. This ensures that the ideal-model adversary is not activated after a dummy party has sent its input to $[\mathcal{G}]$ *before* $[\mathcal{G}]$ receives this input. Note that he would otherwise be able to learn or modify a party's input and output through an `online-attack` instruction at a moment when the party has already received its input.

Activation Instructions. In the UC framework, protocol parties are activated via `external write` instructions. This mechanism cannot be applied to parties that are offline, however. For instance, consider a party P that wants to send messages to multiple parties via data diodes while being offline. In order to do so, P must be activated multiple times. This raises a problem since there is no way to activate P via an `external write` instruction.

In order to address this problem, we allow the adversary to send *activation instructions*. Formally, $\mathcal{A}$ may activate a party P by writing $(\texttt{activate}, P)$ on its outgoing message tape. P will then be activated.

5.2.6 Fortified UC Emulation

We now define the execution experiment in our framework by applying the rules specified in Sections 5.2.1 to 5.2.5 to the UC execution experiment:

Definition 5.3 (Fortified UC Execution Experiment)**.** *An execution of a protocol σ with* adversary *$\mathcal{A}$ and an* environment *$\mathcal{Z}$ on input $a \in \{0,1\}^*$ and with security parameter $n \in \mathbb{N}$ is a run of a system of ITMs subject to the following restrictions:*

- *First, $\mathcal{Z}$ is activated on input $a \in \{0,1\}^*$. At each activation, $\mathcal{Z}$ may mark each channel that exists between $\mathcal{Z}$ and a protocol party either* online *or* offline*. In addition, for each channel to $\mathcal{Z}$, $\mathcal{Z}$ is informed upon each activation if it can receive output from that channel (cf. Section 5.2.2).*
- *The first ITI to be invoked by $\mathcal{Z}$ is the adversary $\mathcal{A}$. The corruption model is as specified in Section 5.2.3.*
- *$\mathcal{Z}$ may invoke a single instance of a* challenge protocol*, which is set to be σ by the experiment. The SID of σ is determined by $\mathcal{Z}$ upon invocation.*
- *$\mathcal{Z}$ may provide inputs to the adversary. In addition, $\mathcal{Z}$ may provide inputs to the parties of σ subject to the protocol architecture (cf. Section 5.2.1). (Note that among the parties that may receive input from $\mathcal{Z}$ are interface modules, cf. Section 5.2.4.)*
- *The adversary $\mathcal{A}$ may give subroutine outputs to $\mathcal{Z}$. In addition, $\mathcal{A}$ may send messages to the parties of σ subject to the protocol architecture (Section 5.2.1). At each activation, $\mathcal{A}$ is given the status (cf. Section 5.2.2). Moreover, $\mathcal{A}$ may activate a party through* `activate` *instructions (cf. Section 5.2.5).*
- *Each party of σ may send messages to the adversary, provide inputs to its sub-parties and give subroutine outputs to the parties of which it is a sub-party or to the environment $\mathcal{Z}$ subject to the protocol architecture (Section 5.2.1). Immediate communication (i.e. providing inputs or giving subroutine outputs) may trigger the notify transport mechanism activating the adversary as specified in Section 5.2.5.*
- *The ITIs take turns during the execution experiment, i.e., whenever an ITI writes an allowed external write instruction, then the targeted ITI is activated and the sending ITI is suspended. If an ITI suspends its computation without writing an external write instruction or if it writes a disallowed external write instruction, then the environment $\mathcal{Z}$ is activated.*

(Note that this has the effect that at any point in time throughout the execution experiment only a single ITI is active.)

- *At the end of the execution experiment, $\mathcal{Z}$ outputs a single bit.*

Denote by $\mathrm{Exec}_{\mathrm{FortUC}}(\sigma, \mathcal{A}, \mathcal{Z})(n, a) \in \{0,1\}$ *the output of the environment* $\mathcal{Z}$ *on input* $a \in \{0,1\}^*$ *and with security parameter* $n \in \mathbb{N}$ *when interacting with* σ *and* $\mathcal{A}$ *according to the above definition.*

We now define security in our framework in analogy to the UC framework:

Definition 5.4 (Emulation in the Fortified UC Framework)**.** *Let π and ϕ be protocols. π is said to emulate ϕ in the Fortified UC framework, denoted by $\pi \underset{\#\#}{\geq} \phi$,*[16] *if for every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{S}$ such that for every PPT environment $\mathcal{Z}$ there exists a negligible function* negl *such that for all $n \in \mathbb{N}, a \in \{0,1\}^*$ it holds that*

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

Let π be a protocol with N *main* parties $P_1, \ldots, P_N$ and $L \leq N$. We will later say that "π emulates ϕ for up to L parties under adversarial control" if emulation holds for all (real-model) PPT adversaries $\mathcal{A}$ corrupting at most L parties $P \in \{P_1, \ldots, P_N\}$.

5.3 Properties of the Framework

In this section, we show some important properties of our new security notion.

As in the UC framework (cf. Proposition 2.29), the dummy adversary is also complete in our framework.

Definition 5.5 (Emulation with Respect to the Dummy Adversary)**.** *Define the* dummy adversary $\mathcal{D}$ *as follows: i) When receiving a message* (sid, pid, m) *from the environment* $\mathcal{Z}$*,* $\mathcal{D}$ *sends* m *to the party with extended identity* (pid, sid)*. ii) When receiving* $(\texttt{physical-attack}, \mathcal{M})$ or $(\texttt{online-attack}, P)$ or $(\texttt{taint}, P)$ or $(\texttt{activate}, P)$ from $\mathcal{Z}$, $\mathcal{D}$ carries out that instruction. iii) When receiving m from the party with PID *pid* and SID *sid*, $\mathcal{D}$ sends (sid, pid, m) to $\mathcal{Z}$. iv) When receiving the instruction `status` from $\mathcal{Z}$, $\mathcal{D}$ sends the status $\mathcal{Z}$.

Let π and ϕ be protocols. π is said to emulate ϕ with respect to the dummy adversary *in the Fortified UC framework if there exists a* PPT*-adversary $\mathcal{S}_{\mathcal{D}}$ such that for every* PPT*-environment $\mathcal{Z}$ there exists negligible function* negl *such that for all $n \in \mathbb{N}, a \in \{0,1\}^*$ it holds that*

$$|\mathsf{Pr}[\mathrm{Exec}_{\mathrm{FortUC}}(\pi, \mathcal{D}, \mathcal{Z})(n, a) = 1] - \mathsf{Pr}[\mathrm{Exec}_{\mathrm{FortUC}}(\phi, \mathcal{S}_{\mathcal{D}}, \mathcal{Z})(n, a) = 1]| \leq \mathsf{negl}(n)$$

Proposition 5.6 (Completeness of the Dummy Adversary)**.** *Let π and ϕ be protocols. Then, $\pi \underset{\#\#}{\geq} \phi$ if and only if π emulates ϕ with respect to the dummy adversary in the Fortified UC framework.*

[16]Think of "##" as a fence, i.e. part of a fortification.

Proof Idea. The proof is almost identical to the proof in the UC framework (cf. [Can01] or the proof of Proposition 4.8 in this work). The main difference is that the environment $\mathcal{Z}_{\mathcal{A}}$, which internally runs a copy of a given adversary $\mathcal{A}$ and environment $\mathcal{Z}$, forwards the status to $\mathcal{A}$ each time $\mathcal{A}$ is activated in $\mathcal{Z}_{\mathcal{A}}$'s internal simulation. Note that $\mathcal{Z}_{\mathcal{A}}$ can obtain the status by sending `status` to the dummy adversary $\mathcal{D}$. $\square$

Like UC emulation (cf. Proposition 2.30), emulation in the Fortified UC framework is transitive.

Proposition 5.7 (Transitivity). *Let π_1, π_2, π_3 be protocols. If $\pi_1 \underset{\#\#}{\geq} \pi_2$ and $\pi_2 \underset{\#\#}{\geq} \pi_3$ then it holds that $\pi_1 \underset{\#\#}{\geq} \pi_3$.*

Proof Idea. The proof follows from the same argument as in the UC framework (cf. [Can01] or the proof of Proposition 4.9 in this work). $\square$

Our security notion can be shown to be equivalent to adaptive UC security for "plain protocols" that do not have unhackable sub-parties. Before stating the theorem, we first formally define plain protocols:

Definition 5.8 (Plain Protocols). *A protocol π is called* plain *if all of the following hold: i) all (sub-)parties in π are hackable ii) π only uses standard channels iii) π has no interface modules iv) each party in π is combined with all of its sub-parties v) each (sub-)party in π that is not a (dummy) party of an ideal (sub-)protocol is connected to the adversary via a standard channel vi) (sub-)parties in π only make calls to standard functionalities*

Note that plain protocols can be interpreted as protocols in the UC framework in a natural way.

In the constructions presented later in this chapter we will use certain plain protocols that additionally have air-gap switches between specific ITIs as sub-protocols. In the following, we define this type of protocols:

Given a plain protocol protocol π, define $\overline{\pi}$ to be identical to π except for the following: All standard channels, except for the ones between combined parties and between parties and the environment, are replaced with *initially disconnected air-gap switches.* Upon receiving input, each party connects all of its air-gap switches.

We will later use protocols of the form $\overline{\texttt{SC}(\mathcal{F})}$[17] as sub-protocols in our constructions.

We are now ready to state the theorem:

Theorem 5.9 (Equivalence with UC Emulation for Plain Protocols). *Let π and ϕ be plain protocols. Then,*

$$\pi \underset{\#\#}{\geq} \phi \iff \pi \underset{\text{UC}}{\geq} \phi \iff \overline{\pi} \underset{\#\#}{\geq} \overline{\phi},$$

where $\underset{\text{UC}}{\geq}$ denotes UC emulation with respect to adaptive *(PID-wise[18]) corruption.*

[17]Note that in the ideal protocol $\overline{\texttt{SC}(\mathcal{F})}$, each dummy party is connected to $\mathcal{F}$ via an initially disconnected air-gap switch and to the environment via a standard channel.

[18]Recall (cf. Section 5.2.3) that the PIDs of combined parties (which can only be corrupted together in our framework) are equal in their first component. We therefore prove equivalence with a variant of adaptive UC security where parties with PIDs which are equal in their first component are corrupted together (this is what we mean by "PID-wise corruption").

Proof Idea. In the following, we briefly sketch some arguments for the statement $\pi \underset{\text{UC}}{\geq} \phi \implies \pi \underset{\#\#}{\geq} \phi$. The other statements follow from very similar arguments.

Since $\pi \underset{\text{UC}}{\geq} \phi$, it follows that there exists a simulator $\mathcal{S}_{\mathcal{D}}$ for the dummy adversary $\mathcal{D}$. Given $\mathcal{S}_{\mathcal{D}}$, one can construct a simulator $\widetilde{\mathcal{S}}_{\mathcal{D}}$ for the dummy adversary in the Fortified UC framework. $\widetilde{\mathcal{S}}_{\mathcal{D}}$ is defined roughly as follows: $\widetilde{\mathcal{S}}_{\mathcal{D}}$ internally runs $\mathcal{S}_{\mathcal{D}}$, relaying the messages between $\mathcal{S}_{\mathcal{D}}$ and the other ITIs. Upon receiving a `physical-attack` or `online-attack` instruction from $\mathcal{Z}$, $\widetilde{\mathcal{S}}_{\mathcal{D}}$ informs $\mathcal{S}_{\mathcal{D}}$ which parties are to be corrupted and then carries out the respective instruction. When $\mathcal{S}_{\mathcal{D}}$ reports that he has received a message from an ideal functionality, $\widetilde{\mathcal{S}}_{\mathcal{D}}$ first reports a notify transport message to the environment. $\widetilde{\mathcal{S}}_{\mathcal{D}}$ ignores all notify transport messages. Upon receiving `status` from the environment, $\widetilde{\mathcal{S}}_{\mathcal{D}}$ reports that each party is online via all of its channels in π.

Assume for the sake of contradiction that there is a Fortified UC environment $\widetilde{\mathcal{Z}}$ that can distinguish between an interaction with π and $\mathcal{D}$ or ϕ and $\widetilde{\mathcal{S}}_{\mathcal{D}}$ with non-negligible probability. Then one can construct a UC environment $\mathcal{Z}$ that can distinguish between an interaction with π and $\mathcal{D}$ or ϕ and $\mathcal{S}_{\mathcal{D}}$ with non-negligible probability, thus reaching a contradiction. $\mathcal{Z}$ is defined roughly as follows: $\mathcal{Z}$ internally runs $\widetilde{\mathcal{Z}}$, relaying the messages between $\widetilde{\mathcal{Z}}$ and the other ITIs. When $\widetilde{\mathcal{Z}}$ sends a `physical-attack` or `online-attack` instruction, $\mathcal{Z}$ corrupts the respective parties by sending `corrupt` messages to the adversary. When the adversary reports that he has received a message from an ideal functionality, $\mathcal{Z}$ first reports a notify transport message to $\widetilde{\mathcal{Z}}$. Each time $\widetilde{\mathcal{Z}}$ is activated in $\mathcal{Z}$'s internal simualtion, $\mathcal{Z}$ informs $\widetilde{\mathcal{Z}}$ that it can receive output via each of its channels to the parties in π. When $\widetilde{\mathcal{Z}}$ sends `status`, $\mathcal{Z}$ reports to $\widetilde{\mathcal{Z}}$ that each party is online via all of its channels in π.

It is easy to see that the view of $\widetilde{\mathcal{Z}}$ when interacting with the challenge protocol is identical to its view when internally run by $\mathcal{Z}$ (note that the notify transport mechanism is only triggered in a plain protocol if a party sends a message to a standard ideal functionality, which, by convention, immediately notifies the adversary upon receiving input). □

The following theorem guarantees that, like UC security (cf. Theorem 2.31), the security notion of the Fortified UC framework is closed under protocol composition:

Theorem 5.10 (Composition Theorem)**.** *Let π, ϕ, ρ be protocols. Then,*

$$\pi \underset{\#\#}{\geq} \phi \implies \rho^{\pi} \underset{\#\#}{\geq} \rho^{\phi}$$

Sketch. The proof is very similar to the proof of the composition theorem of the UC framework(cf. [Can01]). The two main differences are the following

The environment $\mathcal{Z}_{\pi}$, which internally runs a given environment $\mathcal{Z}$, the protocol ρ and instances of π and ϕ and interacts with the dummy adversary $\mathcal{D}$ and either π or ϕ as challenge protocol, behaves as in proof of the composition theorem of the UC framework (cf. [Can01]) and additionally does the following:

1. $\mathcal{Z}_{\pi}$ marks each channel to a party in the challenge protocol according to the online state of the respective calling party in ρ in its internal simulation. This ensures that the online states of the parties in the challenge protocol when interacting with $\mathcal{Z}_{\pi}$ are the same as when run as subroutines of ρ in an interaction with the environment $\mathcal{Z}$.

2. $\mathcal{Z}_\pi$ determines if a party E in its internal simulation who is a calling party of a party P in the challenge protocol is online via a channel C to P by deriving the relevant information from the status reported by the adversary (which contains information about whether P is online via a channel $C' \neq C$ to another ITM $\mu \neq E$) *and* by checking whether it can receive output[19] via C from P.[20] This ensures that the online state of E when internally run by $\mathcal{Z}_\pi$ is the same as when running in an interaction between ρ and the environment $\mathcal{Z}$.

3. If $\mathcal{Z}$ sends $(\texttt{physical-attack}, \mathcal{M})$ in $\mathcal{Z}_\pi$'s internal simulation, $\mathcal{Z}_\pi$ sends $(\texttt{physical-attack}, \mathcal{M}')$ to $\mathcal{D}$, where the parties in $\mathcal{M}'$ are the main parties of $\mathcal{Z}_\pi$'s challenge protocol who are the respective sub-parties of the parties in $\mathcal{M}$. Furthermore, if $\mathcal{Z}$ sends $(\texttt{online-attack}, P)$ for a party P in $\mathcal{Z}_\pi$'s challenge protocol, $\mathcal{Z}_\pi$ forwards $(\texttt{online-attack}, P)$ to $\mathcal{D}$. Finally, if $\mathcal{Z}$ sends $(\texttt{online-attack}, T)$ for a party T in $\mathcal{Z}_\pi$'s internal simulation, $\mathcal{Z}_\pi$ checks the online state of T in its internal simulation and ignores this instruction if T is offline or internally carries out this instruction if T is online. Furthermore, if T is combined with a party P in $\mathcal{Z}_\pi$'s challenge protocol, $\mathcal{Z}_\pi$ forwards $(\texttt{online-attack}, P)$ to $\mathcal{D}$.

The simulator $\mathcal{S}$, which internally runs copies of the simulator $\mathcal{S}_\pi$ implied by $\pi \underset{\#\#}{\geq} \phi$ and interacts with a given environment $\mathcal{Z}$ and the protocol ρ^ϕ, behaves as in proof of the composition theorem of the UC framework (cf. [Can01]) and additionally does the following:

$\mathcal{S}$ keeps track of a "simulated status" as follows:

1. If E is a party who is *not* a party of an instance of ϕ and C is channel between E and an ITM who is also *not* a party of an instance of ϕ, then E is online via C in $\mathcal{S}$'s "simulated status" if and only if the status $\mathcal{S}$ receives[21] from the experiment states that E is online via C.

2. At each activation, $\mathcal{S}$ internally hands the copies of the simulator $\mathcal{S}_\pi$ a status that is derived from the status that $\mathcal{S}$ receives from the experiment (by taking the information about the channels involving the parties in the respective instance of ϕ). Afterwards, $\mathcal{S}$ sends the instruction `status` to all copies of $\mathcal{S}_\pi$, receiving a status from each copy.

3. If P is a party of an instance of ϕ and C' is a channel between P and any other ITM, then P is online via C' in $\mathcal{S}$'s "simulated status" if and only if the status reported by the respective copy of the simulator $\mathcal{S}_\pi$ for that instance of ϕ claims P to be online via C'.

4. If E is a party who is *not* a party of an instance of ϕ and C'' is channel between E and a party P who is a party of an instance of ϕ, then E is online via C'' in $\mathcal{S}$'s "simulated status" if and only if the status that $\mathcal{S}$

[19]Recall that for each channel to $\mathcal{Z}_\pi$, $\mathcal{Z}_\pi$ is informed upon each activation if it can receive output from that channel, cf. Section 5.2.2.

[20]Recall that, by definition, E is online via channel C if and only if E can receive output via C from the sub-party P and P is online via a channel $C' \neq C$ to another ITM $\mu \neq E$, cf. Section 5.2.2.

[21]Recall that, at each activation, the adversary $\mathcal{S}$ gets informed via which channels each party is online, cf. Section 5.2.2.

receives from the experiment states that E is online via C'' *and* the status reported by the respective copy of $\mathcal{S}_\pi$ for that instance of ϕ claims that P is online via a channel $\widetilde{C} \neq C''$ to another ITM $\mu \neq E$.

When the environment $\mathcal{Z}$ sends the instruction `status` to $\mathcal{S}$, then $\mathcal{S}$ reports the "simulated status" to $\mathcal{Z}$. When $\mathcal{Z}$ sends $(\texttt{online-attack}, E)$ for a party E who is *not* a party of an instance of ϕ, then $\mathcal{S}$ checks the online state of E that is implied by the "simulated status" and ignores this instruction if E is offline or carries out the instruction if E is online. When $\mathcal{Z}$ sends $(\texttt{online-attack}, P)$ for a party P that is a party of an instance of ϕ, then $\mathcal{S}$ forwards this message to the respective copy of $\mathcal{S}_\pi$ for that instance.

□

Theorems 5.9 and 5.10 allow for modular composition with UC-secure protocols. This will be used in the subsequent sections where we will construct protocols $\rho^{\overline{\texttt{SC}(\mathcal{F})}}$ such that $\rho^{\overline{\texttt{SC}(\mathcal{F})}} \underset{\#\#}{\geq} \texttt{AG}([\mathcal{G}])$ for some fortified functionality $[\mathcal{G}]$. Given a protocol π such that $\pi \underset{\text{UC}}{\geq} \texttt{SC}(\mathcal{F})$, we can replace all instances of the ideal protocol $\overline{\texttt{SC}(\mathcal{F})}$ which are called by ρ by instances of the protocol $\overline{\pi}$ as follows: By Theorem 5.9, it holds that $\overline{\pi} \underset{\#\#}{\geq} \overline{\texttt{SC}(\mathcal{F})}$. Hence, we have that $\rho^{\overline{\pi}} \underset{\#\#}{\geq} \rho^{\overline{\texttt{SC}(\mathcal{F})}}$ by Theorem 5.10. Therefore, $\rho^{\overline{\pi}} \underset{\#\#}{\geq} \texttt{AG}([\mathcal{G}])$ by transitivity of Fortified UC emulation (Proposition 5.7).

Technical Remarks: Further Discussion of the Composition Theorem

1. Not giving the environment the possibility to learn if it can receive output via a channel between the environment and a party does not lead to a composable security notion. As an example, consider a two-party protocol π (realizing some non-reactive functionality) that only uses standard channels. In particular, parties P_1, P_2 in π are always online. Consider a protocol π' that is identical to π, except that the parties are connected to the environment via initially connected air-gap switches. Each party in π' disconnects its air-gap switch to the environment upon receiving input. Before giving output, a party connects its air-gap switch to the environment again. It is easy to see that π emulates π' according to this modified notion (that is identical to the security notion of the Fortified UC framework, except that the environment is not informed about whether it can receive output from a channel).

   Now, consider a protocol ρ^π that consists of two parties E_1, E_2 making subroutine calls to one instance of π, i.e. P_i is a sub-party of E_i in ρ^π. Each E_i is offline via all channels, except to P_i. It holds that ρ^π does not emulate $\rho^{\pi'}$. This is because the parties E_i are online in ρ^π but offline in $\rho^{\pi'}$. Hence, an environment who instructs the dummy adversary $\mathcal{D}$ to send an `online-attack` instruction to one of the E_i's can easily distinguish these two protocols by observing if that party becomes corrupted or not.

2. Stipulating that a party is online if it can receive input from the environment, i.e. not giving the environment the possibility to modify the online state of its channels to the parties (by marking the channel), neither leads to a composable security notion. As an example, consider a two-party protocol π where each party P_i is connected to the environment via a standard channel. Furthermore, both parties are connected to the adversary via initially connected air-gap switches. Let π' be identical to π, except that all air-gap switches to the adversary are initially disconnected. It is easy to see that π emulates π' according to this modified notion (that is identical to the security notion of the Fortified UC framework, except that a party is online if it can receive input from the environment).

   Now, consider a protocol ρ^π that consists of two parties E_1, E_2 making subroutine calls to one instance of π, i.e. P_i is a sub-party of E_i in ρ^π. Each E_i is offline via all channels, except to P_i. By construction, the parties P_i are still initially online in ρ^{π}. However, they are initially offline in $\rho^{\pi'}$. Hence, an environment who instructs the dummy adversary $\mathcal{D}$ to send an `online-attack` instruction to one of the P_i's can easily distinguish these two protocols by observing if that party becomes corrupted or not.

3. Only giving the adversary the current online state of a party instead of the information via which channels a party is online does also not lead to a composable security notion. As an example, consider a two-party protocol π where the environment is connected to the parties P_1, P_2 via initially connected air-gap switches (i.e. the environment operates these air-gap switches). Furthermore, both parties in π are connected to the adversary via initially disconnected air-gap switches. Let π' be identical to π, except that all air-gap switches to the adversary are initially connected. It is easy to see that π emulates π' according to this modified notion (that is identical to the security notion of the Fortified UC framework, except that the adversary is only given the current online state of each party).

   Now, consider a protocol ρ^{π} that consists of two parties E_1, E_2 making subroutine calls to one instance of π, i.e. P_i is a sub-party of E_i in ρ^{π}. Each E_i is (only) connected to the environment and adversary via initially connected air-gap switches. On any input, each E_i disconnects its air-gap switch to the environment. On input 0, each E_i disconnects its air-gap switch to the adversary but lets its air-gap switch to P_i remain connected. In contrast, on input 1, each E_i disconnects its air-gap switch to P_i but lets its air-gap switch to the adversary remain connected.

   It holds that ρ^{π} does not emulate $\rho^{\pi'}$. This can be argued as follows: Consider the environment $\mathcal{Z}$ interacting with the dummy adversary $\mathcal{D}$ that randomly chooses a bit b, hands b to E_1 as input, and then instructs $\mathcal{D}$ to send an `online-attack` instruction to E_1. By construction, the party E_1 will then be corrupted or not depending on the input b. More specifically, E_1 will be corrupted if $b = 1$ and remain uncorrupted otherwise (since P_1 in π is offline). However, in the protocol $\rho^{\pi'}$, E_1 is always online regardless of its inputs. This is because E_1 is either online via its channel to the adversary or to the party in π' who, by construction, is always online. Therefore, a potential simulator interacting with $\rho^{\pi'}$ who only gets the online state of the parties cannot decide if the online attack on E_1 should be carried out or ignored.

5.4 Construction for Non-Reactive Functionalities

In this section, we present a construction for realizing the fortified functionality of every *non-reactive* (standard adaptively well-formed[22]) ideal functionality.

The broad idea is to have the parties $P_1, \ldots, P_N$ send *encrypted shares* of their inputs via *data diodes* in an *offline sharing phase* and subsequently use these shares to compute the desired function in an *online compute phase*. This, however, cannot be done straightforwardly. To begin with, the parties are not able to retrieve public keys themselves in the sharing phase since this would necessitate going online, making them susceptible to online attacks. Therefore, each party P_i sends its shares to an unhackable sub-party called *encryption unit* (Enc-unit) via a data diode. The Enc-unit retrieves the public keys and sends encrypted shares to hackable sub-parties of the designated receivers called *buffers* (note that since the parties $P_1, \ldots, P_N$ are offline they are unable to receive messages).

Furthermore, each message has to be authenticated so that the adversary cannot change the input of a party by modify the messages it sends. One could do this with an additional unhackable "authentication unit" which signs each ciphertext or have the Enc-unit sign all ciphertexts. However, since we

[22]Cf. Section 2.4.2 for a definition of standard adaptively well-formed.

want to use as few and as simple unhackable sub-parties as possible, we take a different approach. Each party P_i sends its shares together with valid *signatures* to its Enc-unit. The verification key is sent, over an intermediary sub-party called *join* (J), to a hackable sub-party called *registration module* (RM) which disconnects itself from J after receiving input and forwards the verification key to a *public bulletin board* via a data diode. Once a party P_i has sent all of its shares, it erases everything, except for its own share, its verification key and its decryption key. In order for this sign-then-encrypt approach to be secure, we assume that the PKE scheme is non-malleable (*NM-CPA-secure*) and that the digital signature is unforgeable (*EUF-naCMA secure*) and also satisfies a property we call *length-normal*, guaranteeing that signatures of messages of equal length are also of equal length. This prevents an adversary from learning information of plaintexts based on the length of their ciphertext. Each party P_i is connected to its sub-party J via an *initially disconnected* air-gap switch in order to prevent the adversary from corrupting P_i's RM (or J) but *not* P_i before P_i has received its input.

In the compute phase, the adversary must be prevented from using values that are *different* from the shares sent by the honest parties to the corrupted parties in the sharing phase. Otherwise, he would be able to modify the inputs of the parties who were honest in the sharing phase. The parties P_i therefore not only use the shares they received but also the signatures of these shares and the registered verification keys during the compute phase. The result of the compute phase is the error symbol $\perp$ if not all signatures are valid. Since the signing keys were erased at the end of the sharing phase, the adversary cannot generate new valid signatures for parties P_i corrupted after receiving input. He is also unable to revoke the verification key of such parties since this would require corrupting the respective RM, which is impossible since that party is offline.

Moreover, an adversary could *swap* a message in the sharing phase addressed to (the buffer of) an honest party P_j with a ciphertext of a share and signature received by a corrupted party (by encrypting that tuple with the respective public key). Furthermore, an adversary controlling at least two parties P_i, P_j knows two shares and valid signatures of each party and could use one of these tuples *twice* in the compute phase. To prevent these attacks, a party P_i signs each share *along with the designated receiver's PID*. In addition, a party P_i also includes its own PID in each message it sends in order to prevent the adversary from reusing messages sent by honest parties for the parties corrupted before receiving input.

Finally, one cannot simply send the result of the compute phase to a party P_i since this would allow the adversary to learn and modify the output of the parties corrupted after receiving input. Instead, we introduce another unhackable sub-party called *output interface module* (OIM). Each party P_i sends not only the shares of its input x_i but also shares of a *random pad* r_i and of a *MAC key* k_i in the sharing phase. Furthermore, each party P_i sends r_i and k_i to its OIM via a data diode. In the compute phase, the parties will then use these shares to compute $(y_i + r_i, \mathrm{Mac}(k_i, y_i + r_i))$, where y_i is the desired output value (of party P_i). Each party then sends its result to its OIM, which will check authenticity by verifying the MAC tag and, if correct, reconstruct and output the value y_i.

In the following, we will take a modular approach and define a functionality $\mathcal{F}_\mathcal{G}$ that implements the verification of the input values (in particular, verification of the signatures) in the compute phase as well as the subsequent multi-party computation on the shares. Using Theorems 5.9 and 5.10, we will be able to replace the sub-protocol $\overline{\mathtt{SC}(\mathcal{F}_\mathcal{G})}$ in our construction with an existing adaptively UC-secure protocol (cf. Remark 5.17).

We first define the functionality $\mathcal{F}_\mathcal{G}$.

Construction 3. *Let $\mathcal{G}$ be a* non-reactive *standard adaptively well-formed ideal functionality. $\mathcal{F}_\mathcal{G}$ proceeds as follows, running with parties $P_1, \ldots, P_N$ and adversary $\mathcal{A}$ and parametrized with a digital signature scheme* SIG *and a message authentication code* MAC*.*

1. *Initialize the Boolean variable* $\mathtt{verify} = \mathtt{true}$*.*
2. *Upon receiving input from party P_i, store it and send $(\mathtt{received}, P_i)$ to $\mathcal{A}$. Upon receiving $(\mathtt{confirmed}, P_i)$ from $\mathcal{A}$, mark P_i as* input given*.*
3. *Upon receiving $(\mathtt{corrupt}, P_i)$, behave like a standard corruption ideal functionality. In addition, forward this message to $\mathcal{G}$.*
4. *Upon receiving from $\mathcal{A}$ a (modified) input for a party P_l marked as* corrupted*, store that input (if an input has already been stored for P_l then overwrite it) and, if not done yet, mark P_l as* input given*.*

Consistency Check

5. *Once each party has been marked as* input given*, check if each stored input is of the form $\overrightarrow{\mathsf{vk}}_i = (\mathsf{vk}_1^{(i)}, \ldots, \mathsf{vk}_N^{(i)})$, $(s_{ji}, r_{ji}, k_{ji}, \sigma_{ji})$ $(j = 1, \ldots, N)$.*
   - *(i) If no, set* $\mathtt{verify} = \mathtt{false}$*.*
   - *(ii) If yes, check if $\overrightarrow{\mathsf{vk}}_1 = \cdots = \overrightarrow{\mathsf{vk}}_N$.*
     - *(A) If this does not hold, set* $\mathtt{verify} = \mathtt{false}$*.*
     - *(B) Else, set $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n) = (\mathsf{vk}_1^{(1)}, \ldots, \mathsf{vk}_N^{(1)})$. For all $i = 1, \ldots, N$, check if $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathsf{vk}_j, P_i, s_{ji}, r_{ji}, k_{ji}, \sigma_{ji}) = 1$ for all $j = 1, \ldots, N$.*
       - *(a) If this does not hold for every i, j, set* $\mathtt{verify} = \mathtt{false}$*.*
       - *(b) Else, proceed with Item 6.*

Reconstruction and Computation

6. *For each $i = 1, \ldots, N$, compute $x_i = s_{i1} + s_{i2} + \cdots + s_{iN}$, $k_i = k_{i1} + k_{i2} + \cdots + k_{iN}$ and $r_i = r_{i1} + r_{i2} + \cdots + r_{iN}$.*
7. *Internally run $\mathcal{G}$ on input $(x_1, \ldots, x_N)$. Let $(y_1, \ldots, y_N)$ be the output of $\mathcal{G}$. For all $i = 1, \ldots, N$, compute $o_i = y_i + r_i$ and $\theta_i \leftarrow \mathrm{Mac}(k_i, o_i)$.*
8. *If party P_i requests an output, proceed as follows:*
   - *(i) If* $\mathtt{verify} = \mathtt{false}$*, send a private delayed output $\perp$ to P_i.*
   - *(ii) Else, if Item 7 has already been carried out, send a private delayed output (o_i, θ_i) to P_i.*
9. *If $\mathcal{A}$ requests an output for a party P_l marked as* $\mathtt{corrupted}$*, proceed as follows:*
   - *(i) If* $\mathtt{verify} = \mathtt{false}$*, send $\perp$ to $\mathcal{A}$.*
   - *(ii) Else, if Item 7 has already been carried out, send (o_l, θ_l) to $\mathcal{A}$.*

10. *Once all parties are corrupted, send all of the private randomness used so far as well as the private randomness $\mathcal{G}$ sends to $\mathcal{A}$ in this case (note that $\mathcal{G}$ is adaptively well-formed) to the adversary $\mathcal{A}$. (Note that this ensures that $\mathcal{F}_{\mathcal{G}}$ is also adaptively well-formed).*

11. *All other messages between $\mathcal{A}$ and $\mathcal{G}$ are ignored.*

Let $\mathcal{F}_{\mathsf{reg}}$ be the public bulletin board functionality (cf. Definition 2.27). Let PKE $= (\mathrm{Gen}_{\mathrm{PKE}}, \mathrm{Enc}, \mathrm{Dec})$ be a public-key encryption scheme (cf. Definition 2.11), SIG $= (\mathrm{Gen}_{\mathrm{SIG}}, \mathrm{Sig}, \mathrm{Vrfy}_{\mathrm{SIG}})$ a digital signature scheme (cf. Definition 2.15) and MAC $= (\mathrm{Gen}_{\mathrm{MAC}}, \mathrm{Mac}, \mathrm{Vrfy}_{\mathrm{MAC}})$ a message authentication code (cf. Definition 2.19).

Given a *non-reactive* standard adaptively well-formed functionality $\mathcal{G}$, we next define our protocol $\Pi_{\mathcal{G}}$ for realizing the ideal protocol $\mathtt{AG}([\mathcal{G}])$.

Construction 4. *Define the protocol $\Pi_{\mathcal{G}}$ as follows:*

Architecture: *See Fig. 5.4 for a graphical depiction.*

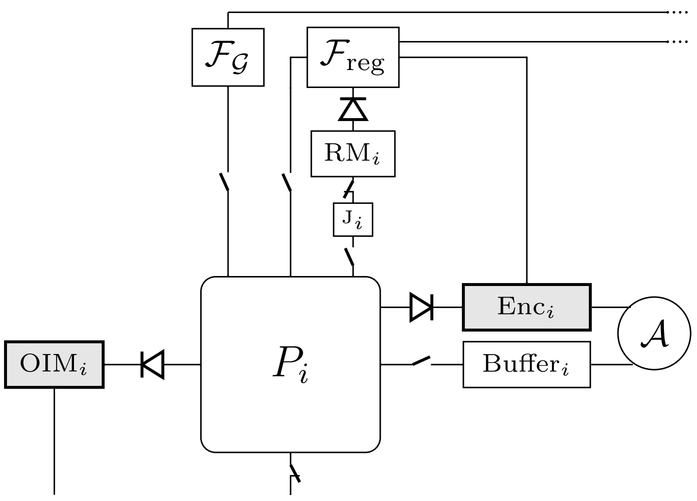


Figure 5.4: Architecture of $\Pi_{\mathcal{G}}$. Each party P_i $(i = 1, \ldots, N)$ has 3 hackable sub-parties, called *buffer*, *registration module* (RM) and *join* (J), and 2 unhackable sub-parties, called Enc(-unit) and OIM. Buffer and Enc-unit are connected to the adversary via standard channels. All air-gap switches, except for P's air-gap switch to the environment and the RM's air-gap switch to J, are initially *disconnected*.

Offline Sharing Phase

Upon input $x_i \in \{0,1\}^p$, each party P_i $(i = 1, \ldots, N)$ does the following:[23]

- Disconnect *air-gap switch to the environment.*
- *Generate* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathrm{Gen}_{\mathrm{PKE}}(1^n)$*,* $k_i \leftarrow \mathrm{Gen}_{\mathrm{MAC}}(1^n)$*,* $(\mathsf{sgk}_i, \mathsf{vk}_i) \leftarrow \mathrm{Gen}_{\mathrm{SIG}}(1^n)$ *and a random pad* $r_i \leftarrow \{0,1\}^q$*.*

[23]Note that, without loss of generality, we assume that all inputs have the same length $p = \mathsf{poly}(n)$ and all outputs have the same length $q = \mathsf{poly}(n)$.

- *Generate shares* $s_{i1} + s_{i2} + \cdots + s_{iN} = x_i$ *and* $k_{i1} + k_{i2} + \cdots + k_{iN} = k_i$ *and* $r_{i1} + r_{i2} + \cdots + r_{iN} = r_i$.
- Connect *air-gap switch to J.*
- *Send* (k_i, r_i) *to* OIM *and* $(\mathsf{pk}_i, \mathsf{vk}_i)$ *to J.*
- *Create signatures* $\sigma_{ij} \leftarrow \mathrm{Sig}(\mathsf{sgk}_i, P_j, s_{ij}, r_{ij}, k_{ij})$ $(j = 1, \ldots, N)$
- *Send* $(P_j, s_{ij}, r_{ij}, k_{ij}, \sigma_{ij})$ $(j \in \{1, 2, \ldots, m\} \setminus \{i\})$ *to* Enc-*unit*
- Erase *everything, except for* $(s_{ii}, r_{ii}, k_{ii}, \sigma_{ii})$, vk_i *and* sk_i.

Registration module and J: *On input* $(\mathsf{pk}_i, \mathsf{vk}_i)$ *to J, J forwards the input to RM. RM then* disconnects *air-gap switch to J and registers* pk_i *and* vk_i *by sending these keys to the public bulletin board functionality* $\mathcal{F}_{\mathsf{reg}}$.

Enc-unit: *Receive a list* $L = \{(P_j, v_j)\}_{j=\{1,\ldots,N\}\setminus\{i\}}$ *from one's main party* P_i. *At each activation, for each* $(P_j, v_j) \in L$, *request* pk_j *belonging to* P_j *from* $\mathcal{F}_{\mathsf{reg}}$. *If retrievable, compute* $c_{ij} \leftarrow \mathrm{Enc}(\mathsf{pk}_j, v_j)$, *send* (P_i, c_{ij})[24] *to buffer of* P_j *and delete* (P_j, v) *from L. Then, go into idle mode.*

Buffer: *Store each message received. On input* `retrieve`, *send all stored messages to one's main party.*

Online Compute Phase

Having completed its last step in the sharing phase, a party P_i *does the following:*

- Connect *air-gap switches to buffer, to* $\mathcal{F}_{\mathsf{reg}}$ *and to* $\mathcal{F}_{\mathcal{G}}$.
- *Request from* $\mathcal{F}_{\mathsf{reg}}$ *all verification keys* $\{\mathsf{vk}_l\}_{l\in\{1,\ldots,N\}\setminus\{i\}}$ *registered by the other parties' registration modules. If not all verification keys can be retrieved yet, go into idle mode and request again at the next activation.*
- *Send* `retrieve` *to buffer and check if the buffer sends at least* $N-1$ *messages. If no, go into idle mode and when activated again send* `retrieve` *and check again. If yes, check if one has received from each party* P_j *a set* $\mathcal{M}_j = \{(P_j, \tilde{c})\}$ *with the following property* $(*)$ (Validity Check)*:*
  There exists a tuple $(P_j, \hat{s}_{ji}, \hat{r}_{ji}, \hat{k}_{ji}, \hat{\sigma}_{ji})$ *and a* $(P_j, c) \in \mathcal{M}_j$ *such that:*
  - $\mathrm{Dec}(\mathsf{sk}_i, c) = (P_j, \hat{s}_{ji}, \hat{r}_{ji}, \hat{k}_{ji}, \hat{\sigma}_{ji})$ *and* $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathsf{vk}_j, P_i, \hat{s}_{ji}, \hat{r}_{ji}, \hat{k}_{ji}, \hat{\sigma}_{ji}) = 1$
  - *For all* $(P_j, \tilde{c}) \in \mathcal{M}_j$ *it holds that either* $\mathrm{Dec}(\mathsf{sk}_i, \tilde{c}) = (P_j, \hat{s}_{ji}, \hat{r}_{ji}, \hat{k}_{ji}, \hat{\sigma}_{ji})$ *or* $(P_j, \tilde{c})$ *is* "invalid", *i.e., either decrypts to a tuple* $(P_j, \tilde{s}_{ji}, \tilde{r}_{ji}, \tilde{k}_{ji}, \tilde{\sigma}_{ji})$ *such that* $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathsf{vk}_j, P_i, \tilde{s}_{ji}, \tilde{r}_{ji}, \tilde{k}_{ji}, \hat{\sigma}_{ji}) = 0$, *or decrypts to a tuple* $(P', \tilde{s}_{ji}, \tilde{r}_{ji}, \tilde{k}_{ji}, \tilde{\sigma}_{ji})$ *such that* $P' \neq P_j$, *or does not decrypt correctly.*

  If this does not hold, send $\perp$ *to* $\mathcal{F}_{\mathcal{G}}$. *Else, send all retrieved verification keys* $(\mathsf{vk}_1, \ldots, \mathsf{vk}_N)$ *as well as all tuples* $(\hat{s}_{ji}, \hat{r}_{ji}, \hat{k}_{ji}, \hat{\sigma}_{ji})$ $(j \in \{1, \ldots, N\})$ *to* $\mathcal{F}_{\mathcal{G}}$.

[24]Sending the sender's PID as prefix is not necessary but simplifies the discussion. Note that for (P_i, c) we also say that "c is addressed as coming from party P_i".

Online Output Phase
Having completed its last step in the compute phase, a party P_i requests output from $\mathcal{F}_{\mathcal{G}}$ and forwards that output to OIM.

OIM: *Store the first input (k_i, r_i) from one's main party. On second input (o_i, θ_i) or $\perp$ from one's main party, do the following: If the received value equals $\perp$, output $\perp$. Otherwise, check if $\mathrm{Vrfy}_{\mathrm{MAC}}(k_i, o_i, \theta_i) = 1$ and output $y_i = o_i + r_i$ if this holds, and $\perp$ otherwise.*

Remark 5.11. *Note that we do not model how to reuse modules such as the registration modules that stay disconnected throughout the protocol execution. In practice, one may assume, e.g., a physical reset mechanism for these modules.*

We will prove that $\Pi_{\mathcal{G}}$ emulates the ideal protocol $\mathtt{AG}([\mathcal{G}])$ (cf. Section 5.2.4 for a definition of $\mathtt{AG}([\mathcal{G}])$) in the Fortified UC framework for adversaries corrupting at most $N - 1$ parties $P \in \{P_1, \dots, P_N\}$ under the assumptions that PKE is NM-CPA-secure (cf. Definition 3.3), SIG is EUF-naCMA-secure and length-normal (cf. Definitions 2.16 and 2.18) and MAC is EUF-1-CMA-secure (cf. Definition 2.20).

Remark 5.12 (Why IND-CPA security is not sufficient)**.**
The following example is inspired by [LK14].

Let $\mathrm{PKE} = (\mathrm{Gen}_{\mathrm{PKE}}, \mathrm{Enc}, \mathrm{Dec})$ *be an* additively homomorphic *IND-CPA secure public key encryption scheme with message space $\{0,1\}^n$ (e.g. the Goldwasser-Micali scheme [GM82]) and* $\mathrm{SIG} = (\mathrm{Gen}_{\mathrm{SIG}}, \mathrm{Sig}, \mathrm{Vrfy}_{\mathrm{SIG}})$ *any digital signature scheme.*

Consider the following encoding $\mathrm{Encode}(m)$*: Any* 0 *in m is transformed to* 00 *and any* 1 *is transformed to* 01 *or* 10*. The inverse* $\mathrm{Decode}(m)$ *of this encoding parses the encoded message as pairs of bits and then maps* 00 *to* 0 *and* 01 *or* 10 *to* 1*. If* 11 *is encountered, the enitre message is be decoded to $\perp$.*

Let $\mathrm{Enc}'(\mathsf{pk}, m) = \mathrm{Enc}(\mathsf{pk}, \mathrm{Encode}(m))$ *and* $\mathrm{Dec}'(\mathsf{sk}, c) = \mathrm{Decode}(\mathrm{Dec}(\mathsf{sk}, c))$*. Note that* $\mathrm{PKE}' = (\mathrm{Gen}_{\mathrm{PKE}}, \mathrm{Enc}', \mathrm{Dec}')$ *is also IND-CPA-secure.*

Consider the following attack: Given a ciphertext $c \leftarrow \mathrm{Enc}'(\mathsf{pk}, m, \sigma)$ (where $\sigma \leftarrow \mathrm{Sig}_{sgk}(m)$), compute

$$c' = c + \mathrm{Enc}'(\mathsf{pk}, 110\dots0)$$

By construction, c' is a ciphertext (under $\mathrm{Enc}(\mathsf{pk}, \cdot)$*) of a bitstring that is identical to the encoded message* $\mathrm{Encode}(m)$*, except that the first two (most significant) bits are flipped. It holds that if the first bit of the underlying message m is* 1*, then the modified ciphertext c' is valid since then the first two bits of* $\mathrm{Encode}(m)$ *are* 01 *or* 10 *and the signature σ will still be valid since it is applied to m and not to the encoding of m. On the other hand, if the first bit of m is* 0*, then the modified ciphertext will not be valid since then the first two bits of* $\mathrm{Encode}(m)$ *would be* 00 *and their complement would be* 11*.*

The above attack enables an adversary $\mathcal{A}$ to learn the first bit of a party P_i's input x_i even if he corrupts P_i after *P_i received its input (but does not corrupt all parties) as follows: $\mathcal{A}$ corrupts every other party, except for one party P_j,* *learning $N-1$ shares of x_i. Furthermore, $\mathcal{A}$ carries out the above attack on the ciphertext c_{ij} intended for (the honest party) P_j. $\mathcal{A}$ can then learn the first bit of x_i by observing if the output of $\mathcal{F}_\mathcal{G}$ (to one of the corrupted parties) is $\perp$ or not.*

Before stating the theorem, we define the following auxiliary experiment, which will be used in the proof.

Definition 5.13 (Auxiliary Experiment)**.** *The experiment $\mathsf{Exp}^{\text{aux}}_{\text{PKE,SIG},\mathcal{A}(z)}(n)$ is defined as follows: At the beginning, the experiment generates keys $(\mathsf{pk},\mathsf{sk}) \leftarrow \text{Gen}_{\text{PKE}}(1^n)$ and $(\mathsf{vk},\mathsf{sgk}) \leftarrow \text{Gen}_{\text{SIG}}(1^n)$. On input $1^n, z$ and pk, the adversary $\mathcal{A}$ may then* non-adaptively *send queries to a signing oracle $\mathcal{O}_{\text{Sig}(\mathsf{sgk},\cdot)}$. Afterwards, the experiment sends vk to $\mathcal{A}$. $\mathcal{A}$ may then send a message of the form $(\mathtt{prf_1},\mathtt{prf_2},m)$ to the experiment. The experiment then computes $\sigma \leftarrow \text{Sig}(\mathsf{sgk},\mathtt{prf_2},m)$, $c^* \leftarrow \text{Enc}(\mathsf{pk},\mathtt{prf_1},m,\sigma)$, and sends c^* to $\mathcal{A}$. During the experiment, $\mathcal{A}$ may send a single* parallel *query to a decryption oracle $\mathcal{O}_{\text{Dec}(\mathsf{sk},\cdot)}$. At the end of the experiment, $\mathcal{A}$ sends a tuple (m',σ') to the experiment. The experiment then checks if $\text{Vrfy}_{\text{SIG}}(vk,m',\sigma')=1$ and m' has not been sent to $\mathcal{O}_{\text{Sig}(\mathsf{sgk},\cdot)}$ before. If this holds, the experiment outputs 1 and 0 otherwise.*

An adversary $\mathcal{A}$ is called valid *if his query to the decryption oracle $\mathcal{O}_{\text{Dec}(\mathsf{sk},\cdot)}$ does not contain c^*.*

We have the following lemma.

Lemma 5.14. *If PKE is IND-pCCA-secure*[25] *and SIG EUF-naCMA-secure, then for every valid PPT adversary $\mathcal{A}$, there exists a negligible negl such that for all $n \in \mathbb{N}, z \in \{0,1\}^*$ it holds that*

$$\mathsf{Pr}[\mathsf{Exp}^{\text{aux}}_{\text{PKE,SIG},\mathcal{A}(z)}(n)=1] \leq \mathsf{negl}(n)$$

Sketch. Assume there exists an adversary $\mathcal{A}$ that wins in the experiment $\mathsf{Exp}^{\text{aux}}_{\text{PKE,SIG},\mathcal{A}(z)}(n)$ with non-negligible probability. Since PKE is IND-pCCA-secure, one can replace c^* by $c' \leftarrow \text{Enc}(\mathsf{pk},0^L)$, where $L = |(\mathtt{prf_1},m,\sigma)|$, incurring only a negligible loss in $\mathcal{A}$'s success probability. Then, one can directly construct an adversary $\mathcal{A}'$ out of $\mathcal{A}$ that breaks the EUF-naCMA-security of SIG with non-negligible probability. $\mathcal{A}'$ simply internally simulates the experiment $\mathsf{Exp}^{\text{aux}}_{\text{PKE,SIG},\mathcal{A}(z)}(n)$ for $\mathcal{A}$ using his signing oracle and c' for c^*. Once $\mathcal{A}$ sends a tuple (m,σ) to the experiment $\mathsf{Exp}^{\text{aux}}_{\text{PKE,SIG},\mathcal{A}(z)}(n)$, $\mathcal{A}'$ sends (m,σ) to the EUF-naCMA experiment. $\mathcal{A}'$ then wins in the EUF-naCMA experiment if and only if $\mathcal{A}$ wins in the experiment $\mathsf{Exp}^{\text{aux}}_{\text{PKE,SIG},\mathcal{A}(z)}(n)$. $\square$

We will use the above experiment to show that an environment $\mathcal{Z}$ cannot send “fake messages” (P_i,c') to an honest party P_j addressed as coming from a party P_i that has *not* been corrupted before receiving input such that i) c' was not generated by the Enc-unit of P_i and ii) (P_i,c') is accepted by P_j. Otherwise, one could build a successful adversary $\mathcal{A}$ in $\mathsf{Exp}^{\text{aux}}_{\text{PKE,SIG},\mathcal{A}(z)}(n)$ as follows: $\mathcal{A}$ guesses indices i,j such that $\mathcal{Z}$ sends a fake message (P_i,c') to P_j. $\mathcal{A}$ simulates the protocol execution for $\mathcal{Z}$. For party P_i, $\mathcal{A}$ sends the tuples $(P_l,s_{il},r_{il},k_{il})$ for $l \neq j$ to $\mathcal{O}_{\text{Sig}(\mathsf{sgk},\cdot)}$ and $(\mathtt{P_i},\mathtt{P_j},s_{ij},r_{ij},k_{ij})$ to the experiment, receiving c^*. $\mathcal{A}$ then uses (P_i,c^*) for (P_i,c_{ij}) in its simulation. If $\mathcal{A}$'s guess is correct, $\mathcal{A}$ can

[25]Cf. Definition 3.2 for a formal definition of IND-pCCA security.

decrypt c' using $\mathcal{O}_{\mathrm{Dec}(\mathsf{sk},\cdot)}$, obtaining a message (P_i, m', σ'). $\mathcal{A}$ can then send (P_j, m', σ') to the experiment and wins because he has never sent a message of the form $(\mathbf{P_j}, m)$ to $\mathcal{O}_{\mathrm{Sig}(\mathsf{sgk},\cdot)}$. Note that if $\mathcal{A}$ had also sent $(P_j, s_{ij}, r_{ij}, k_{ij})$ to $\mathcal{O}_{\mathrm{Sig}(\mathsf{sgk},\cdot)}$, then he would not win if c' decrypts to the same plaintext as c^*, which happens if $\mathcal{Z}$ manages to break the non-malleability of PKE.

Next, we define the simulator for the dummy adversary.

Definition 5.15 (Definition of the Simulator). *Define the simulator* Sim *interacting with an environment $\mathcal{Z}$ and a fortified ideal functionality $[\mathcal{G}]$ as follows:*

1. *At the beginning,* Sim *internally defines N parties corresponding to the parties in $\Pi_{\mathcal{G}}$. Throughout the simulation,* Sim *will keep track of the online state of these parties by marking them as* online *or* offline*. At the beginning,* Sim *marks these parties according to the initial online states of the dummy parties in the ideal protocol (which depend on how $\mathcal{Z}$ has initially marked its channels to these parties).*

2. Sim *initializes a Boolean variable* `verify` = `true`.

3. Sim *carries out the* `physical-attack` *instruction received from $\mathcal{Z}$ on its first activation.* Sim *carries out an* (***online-attack***, P_i) *instruction only if* Sim *has marked party P_i as* online.

4. *Upon receiving* ***status*** *from $\mathcal{Z}$,* Sim *reports the status as follows: For an* honest *party,* Sim *reports that the party is online via its channel to $\mathcal{Z}$ if the party is marked as* online *and has not received input yet. If it has already received input,* Sim *reports that it is online via its channel to its buffer, to $\mathcal{F}_{\mathsf{reg}}$ and to $\mathcal{F}_{\mathcal{G}}$. For a party corrupted via a* ***physical-attack*** *instruction,* Sim *reports that it is online via all of its channels in the real protocol. For a party corruped via an* ***online-attack*** *instruction,* Sim *reports that it is online via the air-gap switches which $\mathcal{Z}$ has instructed to connect.*

5. *Throughout the simulation,* Sim *reports the respective notify transport tokens to $\mathcal{Z}$ (note that we will not mention them anymore in the following).*

6. Sim *generates* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathrm{Gen}_{\mathrm{PKE}}(1^n)$, $k_i \leftarrow \mathrm{Gen}_{\mathrm{MAC}}(1^n)$ *and* $(\mathsf{sgk}_i, \mathsf{vk}_i) \leftarrow \mathrm{Gen}_{\mathrm{SIG}}(1^n)$ *for each party P_i that is* not *corrupted before receiving input (i.e. for each party P_i for which* Sim *has not sent a* (***physical-attack***, $\mathcal{M}$) *instruction such that $P_i \in \mathcal{M}$ and has not sent an* (***online-attack***, P_i) *instruction before P_i received its input).*

7. *For each i such that party P_i is honest,* Sim *reports* (`registered`, sid', RM_i, $\mathsf{pk}_i, \mathsf{vk}_i$) *to $\mathcal{Z}$. If $\mathcal{Z}$ answers with "ok",* Sim *stores* $(\mathsf{pk}_i, \mathsf{vk}_i)$ *as "registered".*

8. *Upon notification by $[\mathcal{G}]$ that an (*honest*) party P_i has sent its input,* Sim *marks P_i as* offline *and generates N random strings $s'_{i1}, \ldots, s'_{iN}$ of length p, N random strings $r'_{i1}, \ldots, r'_{iN}$ of length q and N random strings $k'_{i1}, \ldots, k'_{iN}$ of length $|k_i|$.* Sim *then computes* $\sigma'_{ij} \leftarrow \mathrm{Sig}(\mathsf{sgk}_i, P_j, s'_{ij}, r'_{ij}, k'_{ij})$ *and* $c_{ij} \leftarrow \mathrm{Enc}(\mathsf{pk}_j, P_i, s'_{ij}, r'_{ij}, k'_{ij}, \sigma'_{ij})$ $(j = 1, \ldots, N)$*. Each time $\mathcal{Z}$ instructs to activate the* Enc*-unit of P_i,* Sim *reports the respective tuple (P_i, c_{ij}) to $\mathcal{Z}$ if* pk_j *is stored as "registered".*

9. *Once* Sim *has reported all* (P_i, c_{ij}) $(j = 1, \ldots, N)$ *as well as* $(\texttt{registered}, sid', RM_i, \mathsf{pk}_i, \mathsf{vk}_i)$ *for an honest party* P_i, Sim *marks* P_i *as* online.

10. *If a party* P_i *is corrupted after receiving input,* Sim *sends* $(s'_{ii}, r'_{ii}, k'_{ii}, \sigma'_{ii}, \mathsf{vk}_i, \mathsf{sk}_i)$ *to* $\mathcal{Z}$.

11. *If* $\mathcal{Z}$ *instructs to send a tuple* $(\mathsf{pk}_l, \mathsf{vk}_l)$ *to* $\mathcal{F}_{\mathsf{reg}}$ *to the RM of a party* P_l *corrupted before receiving input,* Sim *stores* $(\mathsf{pk}_l, \mathsf{vk}_l)$ *as "registered" (if such a tuple has already been stored for* P_l *then* Sim *overwrites it).*

12. *Each time* $\mathcal{Z}$ *sends a message addressed to buffer of a party* P_i, Sim *stores that message as a message "received by* P_i*".*

13. *If* $\mathcal{Z}$ *instructs to activate an honest party* P_j *who is marked as* online *and has received at least* $N-1$ *messages and* N *verification keys* vk_l *are stored as "registered", then* Sim *stores* $\overrightarrow{\mathsf{vk}}_j = (\mathsf{vk}_1, \ldots, \mathsf{vk}_N)$ *and reports* $(\texttt{received}, P_i)$ *to* $\mathcal{Z}$. *Upon receiving* $(\texttt{confirmed}, P_i)$ *from* $\mathcal{Z}$, Sim *marks* P_j *as* input given.

14. *If* $\mathcal{Z}$ *sends a tuple consisting of a vector* $\overrightarrow{\mathsf{vk}}_j$ *and* $(s'_{lj}, r'_{lj}, k'_{lj}, \sigma'_{lj})$ $(l = 1, \ldots, N)$ *as the input to* $\mathcal{F}_{\mathcal{G}}$ *for a corrupted party* P_j, *then* Sim *stores that input (if an input has already been stored for* P_j *then* Sim *overwrites it) and, if not done yet, marks* P_j *as* input given.

15. *Once all parties are marked as* input given, Sim *does the following:*

    (i) Sim *checks if* $\overrightarrow{\mathsf{vk}}_1 = \cdots = \overrightarrow{\mathsf{vk}}_N$. *If not,* Sim *sets* $\texttt{verify} = \texttt{false}$.

    (ii) For each j *such that party* P_j *is honest,* Sim *checks if the following two conditions hold:*

    - P_j *has received for each* i *such that party* P_i *was* not *corrupted before receiving input the tuple* (P_i, c_{ij}), *where* c_{ij} *is the respective ciphertext generated by* Sim *in Item 8.*
    - P_j *has received for each* l *such that party* P_l *was corrupted before receiving input a set* $\mathcal{M}_l$ *fulfilling property* $(*)$ *(*Validity Check, *see Page 122, third item in "Online Compute Phase").*

    If at least one of these two conditions does not hold, Sim *sets* $\texttt{verify} = \texttt{false}$.

    (iii) For each tuple consisting of a vector $\overrightarrow{\mathsf{vk}}_j$ *and* $(s'_{lj}, r'_{lj}, k'_{lj}, \sigma'_{lj})$ $(l = 1, \ldots, N)$ *which was stored by* Sim *as the input to* $\mathcal{F}_{\mathcal{G}}$ *for a corrupted party* P_j, Sim *checks the following:*

    - *for each* i *such that party* P_i *was* not *corrupted before receiving input,* Sim *checks if* $(s'_{ij}, r'_{ij}, k'_{ij}) = (s_{ij}, r_{ij}, k_{ij})$, *where* (s_{ij}, r_{ij}, k_{ij}) *is the respective tuple generated in Item 8. If this does not hold or* $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathsf{vk}_i, P_j, s'_{ij}, r'_{ij}, k'_{ij}, \sigma'_{ij}) = 0$, Sim *sets* $\texttt{verify} = \texttt{false}$.
    - *for each* l *such that party* P_l *was corrupted before receiving input,* Sim *sets* $\texttt{verify} = \texttt{false}$ *if* $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathsf{vk}_l, P_j, s'_{lj}, r'_{lj}, k'_{lj}, \sigma'_{lj}) = 0$.

16. Sim extracts *the input, MAC key and random pad of each party* P_l corrupted before receiving input *by (a) decrypting all ciphertexts addressed as coming*

from P_l which are sent by $\mathcal{Z}$ to the buffers of honest parties (using the decryption keys generated in Item 6) and examining the respective plaintexts as described in Item 15 (ii) (second condition)[26]*, and (b) using the shares $\mathcal{Z}$ sends to the* Enc*-unit of P_l (if P_l was corrupted through an* `online-attack` *instruction before receiving its input), and (c) using the inputs $\mathcal{Z}$ sends to $\mathcal{F}_\mathcal{G}$ for corrupted parties in Item 14*[27]*.* Sim *sends each extracted input to $[\mathcal{G}]$.*

17. *Once all parties are marked as* input given *and $\mathcal{Z}$ instructs to activate an honest party P_i, then*

    (i) If $\texttt{verify} = \texttt{true}$*,* Sim *instructs $[\mathcal{G}]$ to send the output to the dummy OIM of P_i.*

    (ii) If $\texttt{verify} = \texttt{false}$*,* Sim *instructs $[\mathcal{G}]$ to output $\perp$ to the dummy OIM of P_i.*

18. *Once all parties are marked as* input given *and $\mathcal{Z}$ requests the output of $\mathcal{F}_\mathcal{G}$ for a party P_i* corrupted after receiving input*, then*

    (i) If $\texttt{verify} = \texttt{true}$*,* Sim *generates a random string $\tilde{y}_i \leftarrow \{0,1\}^q$ and sends $(\tilde{y}_i, \mathrm{Mac}(k_i, \tilde{y}_i))$ to $\mathcal{Z}$, where k_i is the MAC generated by* Sim *in Item 6.*

    (ii) If $\texttt{verify} = \texttt{false}$*,* Sim *sends $\perp$ to $\mathcal{Z}$.*

19. *If $\mathcal{Z}$ sends a message (m', t') addressed to OIM of a party P_i* corrupted after receiving input*, then*

    (i) If $\mathcal{Z}$ has not yet requested the output of $\mathcal{F}_\mathcal{G}$ for P_i yet, Sim *instructs $[\mathcal{G}]$ to output $\perp$ to the dummy OIM of P_i.*

    (ii) If $\mathcal{Z}$ has already requested the output of $\mathcal{F}_\mathcal{G}$ for P_i and Sim *sent $(\tilde{y}_i, \mathrm{Mac}(k_i, \tilde{y}_i))$ (in Item 18) to $\mathcal{Z}$, then*

    - *If $m' \neq \tilde{y}_i$,* Sim *instructs $[\mathcal{G}]$ to output $\perp$ to the dummy OIM of P_i.*
    - *If $m' = \tilde{y}_i$ and $\mathrm{Vrfy}_{\mathrm{MAC}}(k_i, m', t') = 1$, then* Sim *instructs $[\mathcal{G}]$ to send the output to the dummy OIM of party P_i. Otherwise,* Sim *instructs $[\mathcal{G}]$ to output $\perp$ to the dummy OIM of P_i.*

    (iii) If $\mathcal{Z}$ has already requested the output of $\mathcal{F}_\mathcal{G}$ for P_i and Sim *sent $\perp$ (in Item 18) to $\mathcal{Z}$, then* Sim *instructs $[\mathcal{G}]$ to output $\perp$ to the dummy OIM of P_i.*

20. *Once all parties are marked as* input given *and $\mathcal{Z}$ requests the output of $\mathcal{F}_\mathcal{G}$ for a party P_i* corrupted before receiving input*, then*

[26]Note that if a set $\mathcal{M}_l$ fulfills property $(*)$, then there exists a plaintext $(P_l, \hat{s}_{li}, \hat{r}_{li}, \hat{k}_{li}, \hat{\sigma}_{li})$ such that at least one ciphertext in $\mathcal{M}_l$ decrypts to this plaintext and each ciphertext in $\mathcal{M}_l$ either decrypts to this plaintext or is invalid. If this plaintext exists, Sim uses it for reconstructing P_l's input, MAC key and random pad. Note that if this plaintext does not exist, then it holds that $\texttt{verify} = \texttt{false}$.

[27]Note that if a party P_i who was honest during the sharing phase is corrupted, Sim does not use the plaintext $(P_l, \hat{s}_{li}, \hat{r}_{li}, \hat{k}_{li}, \hat{\sigma}_{li})$ obtained through (a) or the shares $\mathcal{Z}$ may send to the Enc-unit of P_l addressed to the buffer of P_i (if P_l was corrupted through an $(\texttt{online-attack}, P_l)$ for reconstructing P_l's input, MAC key and random pad, but instead uses the tuple $(s'_{li}, r'_{li}, k'_{li}, \sigma'_{li})$ that $\mathcal{Z}$ sends to $\mathcal{F}_\mathcal{G}$ as input for P_i in Item 14.

(i) *If* `verify` = `true`*,* Sim *sends* $(y_i + r_i, \mathrm{Mac}(k_i, y_i + r_i))$ *to* $\mathcal{Z}$*, where* y_i *is the output of* $[\mathcal{G}]$ *for party* P_i *and* k_i, r_i *are the MAC key and random pad extracted in Item 16.*

(ii) *If* `verify` = `false`*,* Sim *sends* $\perp$ *to* $\mathcal{Z}$*.*

21. Sim *lets* $\mathcal{Z}$ *determine the output of the dummy OIM of each party* corrupted before receiving input.

We now state the theorem:

Theorem 5.16 (Up to $N-1$ Corrupted Parties, Non-Reactive Case)**.** *Let* $\mathcal{G}$ *be a* non-reactive *standard adaptively well-formed*[28] *functionality. Assume* PKE *is NM-CPA-secure and* SIG *is EUF-naCMA-secure and length-normal, and* MAC *is EUF-1-CMA-secure. Then it holds that*

$$\Pi_{\mathcal{G}} \underset{\#\#}{\geq} \mathtt{AG}([\mathcal{G}])$$

for up to $N-1$ *parties under adversarial control.*

Proof. By Proposition 5.6, it suffices to find a simulator for the dummy adversary.

In the following proof, we will consider a sequence of hybrids $\mathrm{H}_0, \dots, \mathrm{H}_4$. Starting from the real protocol $\Pi_{\mathcal{G}}$, we will define ideal protocols that gradually reduce the simulator's abilities (i.e. restrict the set of parties for which he may learn/modify the inputs/outputs). The final hybrid H_4 will be the ideal protocol $\mathtt{AG}([\mathcal{G}])$ and the simulator as defined in Definition 5.15.

Since NM-CPA security is equivalent to IND-pCCA security (cf. Theorem 3.8), we will use the assumption that PKE is IND-pCCA-secure in the following proof.

Let $\mathcal{Z}$ be an environment that instructs $\mathcal{D}$ to corrupt at most $N-1$ parties $P \in \{P_1, \dots, P_N\}$. Let $\mathrm{out}_i(\mathcal{Z})$ be the output of $\mathcal{Z}$ in the hybrid H_i.

In the following, we will say *corrupted before input* and *corrupted after input* for the sake of brevity.

Hybrid H_0**.** Let H_0 be the execution experiment between the environment $\mathcal{Z}$, the dummy adversary $\mathcal{D}$ and the real protocol $\Pi_{\mathcal{G}}$.

Hybrid H_1**.** Let H_1 be the execution experiment between the environment $\mathcal{Z}$, the ideal protocol $\mathtt{AG}(\mathcal{F}_1)$ and the ideal-model adversary Sim_1, where $\mathcal{F}_1$ and Sim_1 are defined as follows: Define $\mathcal{F}_1$ to be identical to $[\mathcal{G}]$, except for the following: $\mathcal{F}_1$ hands the adversary the inputs and outputs of *every* party (honest and corrupted) and allows him to determine the outputs of the dummy OIMs of *all corrupted* parties (i.e. all parties corrupted before *and* after input). Note that, like $[\mathcal{G}]$, $\mathcal{F}_1$ does *not* allow the adversary to modify the inputs of parties corrupted *after* input (unless all parties are corrupted) and also does not allow him to modify the inputs of honest parties.

Define Sim_1 to be like the simulator in Definition 5.15, except for the following:

In Item 8, Sim_1 reports ciphertexts as they are generated in the real protocol (i.e., ciphertexts containing a share of a party's actual input, of a random pad and of a MAC key as well as a signature of these shares). In Item 10, Sim_1 reports the respective shares as they are generated in the real protocol (i.e. a

[28]Cf. Section 2.4.2 for a definition of standard adaptively well-formed ideal functionalities.

share of a party P_i's actual input, of a random pad and of a MAC key) along with a valid signature of these shares and $\mathsf{vk}_i, \mathsf{sk}_i$. In Item 18, if $\texttt{verify} = \texttt{true}$, Sim_1 reports $(y_i + r_i, \mathrm{Mac}(k_i, y_i + r_i))$ to $\mathcal{Z}$, where y_i is the output Sim_1 receives for the respective party from $\mathcal{F}_1$ and k_i, r_i are the MAC key and one-time pad generated in Items 6 and 8. If $\texttt{verify} = \texttt{false}$, Sim_1 reports $\perp$. In Item 19, if $\mathcal{Z}$ sends a message (m', t') addressed to the OIM of a party P_i (corrupted after input), Sim_1 carries out the program of the OIM in the real protocol (using the MAC key and one-time pad generated in Items 6 and 8), thereby computes an output value $y' \in \{0,1\}^q \cup \{\perp\}$ of the OIM, and then instructs $[\mathcal{G}]$ to output y' to the dummy OIM of P_i.

Consider the following events:

Let $\mathbf{E}_{\text{fakemess}}$ be the event that there exists an *honest* party P_j that retrieves a tuple (P_i, c') in its buffer such that party P_i is *not* corrupted before input and (P_i, c') is "valid", i.e. $\mathrm{Dec}(\mathsf{sk}_j, c') = (P_i, s'_{ij}, r'_{ij}, k'_{ij}, \sigma'_{ij})$ and $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathsf{vk}_i, P_j, s'_{ij}, r'_{ij}, k'_{ij}, \sigma'_{ij}) = 1$, but either $c' \neq c_{ij}$ or c_{ij} has not been generated yet (by the Enc-unit of party P_i).

Let $\mathbf{E}_{\text{fakeinp}}$ be the event that $\mathcal{Z}$ sends an input $(s'_{ij}, r'_{ij}, k'_{ij}, \sigma'_{ij})$ for a corrupted party P_j to $\mathcal{F}_{\mathcal{G}}$ such that party P_i is *not* corrupted before input and $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathsf{vk}_i, P_j, s'_{ij}, r'_{ij}, k'_{ij}, \sigma'_{ij}) = 1$, but $(s'_{ij}, r'_{ij}, k'_{ij}) \neq (s_{ij}, r_{ij}, k_{ij})$ (where vk_i and (s_{ij}, r_{ij}, k_{ij}) were generated by P_i).

Let $\mathbf{E} = \mathbf{E}_{\text{fakemess}} \cup \mathbf{E}_{\text{fakeinp}}$. It holds that

$$\mathsf{Pr}[\mathrm{out}_0(\mathcal{Z}) = 1 \wedge \neg\mathbf{E}] = \mathsf{Pr}[\mathrm{out}_1(\mathcal{Z}) = 1 \wedge \neg\mathbf{E}]$$

This is because if $\mathbf{E}_{\text{fakemess}}$ does not occur then a message in the buffer of an honest party P_j that is addressed as coming from a party P_i who was *not* corrupted before input decrypts to a valid message/signature pair if and only if it equals the ciphertext c_{ij} sent by P_i. Moreover, since $\mathbf{E}_{\text{fakeinp}}$ does not occur, it holds that for each corrupted party P_j, if $\mathcal{Z}$ sends a tuple $(s'_{ij}, r'_{ij}, k'_{ij}, \sigma'_{ij})$ as input to $\mathcal{F}_{\mathcal{G}}$ for P_j such that P_i was not corrupted before input, then either $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathsf{vk}_i, P_j, s'_{ij}, r'_{ij}, k'_{ij}, \sigma'_{ij}) = 0$ or $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathsf{vk}_i, P_j, s'_{ij}, r'_{ij}, k'_{ij}, \sigma'_{ij}) = 1$ and $(s'_{ij}, r'_{ij}, k'_{ij}) = (s_{ij}, r_{ij}, k_{ij})$, where vk_i and (s_{ij}, r_{ij}, k_{ij}) were generated by P_i.

Therefore, it holds that

$$|\mathsf{Pr}[\mathrm{out}_0(\mathcal{Z}) = 1] - \mathsf{Pr}[\mathrm{out}_1(\mathcal{Z}) = 1]| \leq \mathsf{Pr}[\mathbf{E}] \leq \mathsf{Pr}[\mathbf{E}_{\text{fakemess}}] + \mathsf{Pr}[\mathbf{E}_{\text{fakeinp}}]$$

Claim 1: $\mathsf{Pr}[\mathbf{E}_{\text{fakemess}}]$ is negligible.

Consider the following adversary $\mathcal{A}$ in the auxiliary experiment $\mathsf{Exp}^{\text{aux}}_{\text{PKE,SIG},\mathcal{A}(z)}(n)$: At the beginning, $\mathcal{A}$ randomly selects a tuple $(i,j) \in \{1,\dots,N\} \times \{1,\dots,N\}$ such that $i \neq j$. $\mathcal{A}$ then simulates hybrid H_0 using the public key pk from the experiment for pk_j in its internal simulation. When $\mathcal{Z}$ gives the party P_i its input x_i, $\mathcal{A}$ generates shares s_{il}, r_{il}, k_{il} of x_i, of a random pad r_i and of a MAC key k_i just like in H_0. $\mathcal{A}$ sends the tuples $(P_l, s_{il}, r_{il}, k_{il})$ for $l \neq j$ to the signing oracle $\mathcal{O}_{\mathrm{Sig}(\mathsf{sgk},\cdot)}$, receiving signatures σ_{il}. After receiving the verification key vk from the experiment, $\mathcal{A}$ uses vk for vk_i in its internal simulation. Using pk, $\mathcal{A}$ encrypts all tuples $(P_i, s_{il}, r_{il}, k_{il}, \sigma_{il})$ $(l \notin \{i,j\})$ and sends them to the respective party in its internal simulation. Once the message (P_i, c_{ij}) is supposed to be sent in the internal simulation, $\mathcal{A}$ sends $(\mathtt{P_i}, \mathtt{P_j}, s_{ij}, r_{ij}, k_{ij})$ to the experiment, receiving c^*. $\mathcal{A}$ then uses (P_i, c^*) for (P_i, c_{ij}) in its simulation. When P_j is activated and is online and has received at least $N-1$ messages, $\mathcal{A}$ sends all ciphertexts addressed as coming from P_i such that $c \neq c^*$ to the decryption oracle $\mathcal{O}_{\mathrm{Dec}(\mathsf{sk},\cdot)}$ (if c^* has not been generated yet, $\mathcal{A}$ sends all ciphertexts addressed as coming from P_i). For each message (P_l, m, σ) he receives from the oracle $\mathcal{O}_{\mathrm{Dec}(\mathsf{sk},\cdot)}$, $\mathcal{A}$ checks if $\mathrm{Vrfy}_{\mathrm{SIG}}(vk, P_j, m, \sigma) = 1$. If this holds for a message (P_l, m', σ'), then $\mathcal{A}$ sends this message (P_j, m', σ') to the experiment. If during the simulation, P_i is corrupted before input or P_j is corrupted (before or after input) or if no message $\mathcal{A}$ receives from $\mathcal{O}_{\mathrm{Dec}(\mathsf{sk},\cdot)}$ is valid, then $\mathcal{A}$ sends $\perp$ to the experiment.

By construction, it holds that if $\mathbf{E}_{\mathrm{fakemess}}$ occurs and $\mathcal{A}$ has correctly guessed an index (i, j) for which $\mathbf{E}_{\mathrm{fakemess}}$ occurs, then $\mathcal{A}$ sends a message c' to $\mathcal{O}_{\mathrm{Dec}(\mathsf{sk},\cdot)}$ such that $c \neq c^*$ or c^* has not been generated yet and $\mathrm{Dec}(\mathsf{sk}, c') = (P_i, m', \sigma')$ and $\mathrm{Vrfy}_{\mathrm{SIG}}(vk, P_j, m', \sigma') = 1$. Since $\mathcal{A}$ does not send a message of the form $(\mathsf{P_j}, m)$ to the signing oracle $\mathcal{O}_{\mathrm{Sig}(\mathsf{sgk},\cdot)}$, it follows that $\mathsf{Exp}^{\mathrm{aux}}_{\mathrm{PKE,SIG},\mathcal{A}(z)}(n) = 1$. Furthermore, the probability that $\mathcal{A}$ correctly guesses an index (i, j) for which $\mathbf{E}_{\mathrm{fakemess}}$ occurs is at least $1/(N \cdot (N-1))$. Hence,

$$\mathsf{Pr}[\mathsf{Exp}^{\mathrm{aux}}_{\mathrm{PKE,SIG},\mathcal{A}(z)}(n) = 1] \geq \mathsf{Pr}[\mathbf{E}_{\mathrm{fakemess}}]/(N \cdot (N-1))$$

Therefore, since $\mathsf{Pr}[\mathsf{Exp}^{\mathrm{aux}}_{\mathrm{PKE,SIG},\mathcal{A}(z)}(n) = 1]$ is negligible by Lemma 5.14 and $N \cdot (N-1)$ is polynomial in n, it follows that $\mathsf{Pr}[\mathbf{E}_{\mathrm{fakemess}}]$ is also negligible.

Claim 2: $\mathsf{Pr}[\mathbf{E}_{\mathrm{fakeinp}}]$ is negligible.
Consider the following adversary $\mathcal{A}$ against the EUF-naCMA security of SIG: At the beginning, $\mathcal{A}$ randomly selects an index $i \in \{1, \ldots, N\}$. $\mathcal{A}$ then simulates hybrid H_0. When $\mathcal{Z}$ gives the party P_i its input x_i, $\mathcal{A}$ generates shares s_{ij}, r_{ij}, k_{ij} of x_i, of a random pad r_i and of a MAC key k_i just like in H_0. $\mathcal{A}$ sends the tuples $(P_j, s_{ij}, r_{ij}, k_{ij})$ $(j = 1, \ldots, N)$ to the signing oracle $\mathcal{O}_{\mathrm{Sig}(\mathsf{sgk},\cdot)}$, receiving signatures σ_{ij}. After receiving vk, $\mathcal{A}$ uses vk for vk_i, encrypts all tuples $(P_i, s_{ij}, r_{ij}, k_{ij}, \sigma_{ij})$ $(j \neq i)$ and sends them to the respective party in its internal simulation. Each time $\mathcal{Z}$ sends a tuple $(s'_{ij}, r'_{ij}, k'_{ij}, \sigma'_{ij})$ as input for a corrupted party P_j to $\mathcal{F}_{\mathcal{G}}$ such that $(s'_{ij}, r'_{ij}, k'_{ij}) \neq (s_{ij}, r_{ij}, k_{ij})$, $\mathcal{A}$ checks if $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathsf{vk}_i, P_j, s'_{ij}, r'_{ij}, k'_{ij}, \sigma'_{ij}) = 1$. If this holds, $\mathcal{A}$ sends $(P_j, s'_{ij}, r'_{ij}, k'_{ij}, \sigma'_{ij})$ to the experiment. If during the simulation, P_i is corrupted before input or if no message $\mathcal{A}$ checks is valid, then $\mathcal{A}$ sends $\perp$ to the experiment.

By construction, it holds that if $\mathbf{E}_{\mathrm{fakeinp}}$ occurs and $\mathcal{A}$ has correctly guessed an index i for which $\mathbf{E}_{\mathrm{fakeinp}}$ occurs, then $\mathsf{Exp}^{\mathsf{euf\text{-}nacma}}_{\mathrm{SIG},\mathcal{A}(z)}(n) = 1$ because the tuple $(P_j, s'_{ij}, r'_{ij}, k'_{ij}, \sigma_{ij})$ is valid and $(P_j, s'_{ij}, r'_{ij}, k'_{ij}) \neq (P_j, s_{ij}, r_{ij}, k_{ij})$ has not been sent to the signing oracle $\mathcal{O}_{\mathrm{Sig}(\mathsf{sgk},\cdot)}$. Furthermore, the probability that $\mathcal{A}$ correctly guesses an index i for which $\mathbf{E}_{\mathrm{fakeinp}}$ occurs is at least $1/N$. Hence,

$$\mathsf{Pr}[\mathsf{Exp}^{\mathsf{euf\text{-}nacma}}_{\mathrm{SIG},\mathcal{A}(z),}(n) = 1] \geq \mathsf{Pr}[\mathbf{E}_{\mathrm{fakeinp}}]/N$$

Therefore, since $\mathsf{Pr}[\mathsf{Exp}^{\mathsf{euf\text{-}nacma}}_{\mathrm{SIG},\mathcal{A}(z)}(n) = 1]$ is negligible because SIG is EUF-naCMA-secure by assumption and N is polynomial in n, it follows that $\mathsf{Pr}[\mathbf{E}_{\mathrm{fakeinp}}]$ is also negligible.

Hence, there exist a negligible function negl_1 such that

$$|\mathsf{Pr}[\mathrm{out}_0(\mathcal{Z}) = 1] - \mathsf{Pr}[\mathrm{out}_1(\mathcal{Z}) = 1]| \leq \mathsf{negl}_1(n)$$

Hybrid H_2. Let H_2 be the execution experiment between the environment $\mathcal{Z}$, the ideal protocol $\mathtt{AG}(\mathcal{F}_1)$ (again) and the ideal-model adversary Sim_2, where Sim_2 is defined as follows:

Define Sim_2 to be like Sim_1, except for the following: In Item 8, upon notification by $\mathcal{F}_1$ that an (*honest*) party P_i has sent its input, Sim_2 generates N random strings $k'_{i1}, \ldots, k'_{iN}$ of length $|k_i|$ (where k_i is the MAC key generated in Item 6) and computes $\sigma'_{ij} \leftarrow \mathrm{Sig}(\mathsf{sgk}_i, P_j, s_{ij}, r_{ij}, k'_{ij})$ $(j = 1, \ldots, N)$, where the s_{ij} and r_{ij} are still the shares of the input x_i and of a random pad r_i, respectively. Sim_2 then iteratively reports $(P_i, \mathrm{Enc}(\mathsf{pk}_j, P_i, s_{ij}, r_{ij}, k'_{ij}, \sigma'_{ij}))$ $(j \in \{1, \ldots, N\} \setminus \{i\})$ to $\mathcal{Z}$. If P_i is corrupted, Sim_2 sends $(s_{ii}, r_{ii}, k'_{ii}, \sigma'_{ii}, \mathsf{vk}_i, \mathsf{sk}_i)$ to $\mathcal{Z}$ in Item 10. (Note that, in Item 18, Sim_2 still uses the MAC key k_i generated in Item 6 for the output of $\mathcal{F}_{\mathcal{G}}$ to P_i if that output is $\neq \perp$).

Let $\mathrm{H}_{2,0}, \ldots, \mathrm{H}_{2,N}$ be the execution experiment between the environment $\mathcal{Z}$, the ideal protocol $\mathtt{AG}(\mathcal{F}_1)$ and the ideal-model adversary $\mathsf{Sim}_{2,0}, \ldots, \mathsf{Sim}_{2,N}$, respectively, where $\mathsf{Sim}_{2,i}$ is defined as follows:

Define the simulators $\mathsf{Sim}_{2,i}$ to be like Sim_1, except for the following: In Item 8, upon notification by $\mathcal{F}_1$ that an (*honest*) party $P_l \in \{P_1, \ldots, P_i\}$ has sent its input, $\mathsf{Sim}_{2,i}$ generates N random strings $k'_{l1}, \ldots, k'_{lN}$ of length $|k_l|$ (where k_l is the MAC key generated in Item 6), computes $\sigma'_{lj} \leftarrow \mathrm{Sig}(\mathsf{sgk}_l, P_j, s_{lj}, r_{lj}, k'_{lj})$ $(j = 1, \ldots, N)$, and iteratively reports $(P_l, \mathrm{Enc}(\mathsf{pk}_j, P_l, s_{lj}, r_{lj}, k'_{lj}, \sigma'_{lj}))$ $(j \in \{1, \ldots, N\} \setminus \{l\})$ to $\mathcal{Z}$. If a party $P_l \in \{P_1, \ldots, P_i\}$ is corrupted after having received input, $\mathsf{Sim}_{2,i}$ sends $(s_{ll}, r_{ll}, k'_{ll}, \sigma'_{ll}, \mathsf{vk}_l, \mathsf{sk}_l)$ to $\mathcal{Z}$ in Item 10.

It holds that

$$\mathsf{Pr}[\mathrm{out}_{2,0}(\mathcal{Z}) = 1] = \mathsf{Pr}[\mathrm{out}_1(\mathcal{Z}) = 1]$$

and

$$\mathsf{Pr}[\mathrm{out}_{2,N}(\mathcal{Z}) = 1] = \mathsf{Pr}[\mathrm{out}_2(\mathcal{Z}) = 1]$$

Assume that there exists a non-negligible function ϵ such that $|\mathsf{Pr}[\mathrm{out}_1(\mathcal{Z}) = 1] = \mathsf{Pr}[\mathrm{out}_2(\mathcal{Z}) = 1]| > \epsilon$. Then there exists an $i^* \in \{1, \ldots, N\}$ such that

$$|\mathsf{Pr}[\mathrm{out}_{2,i^*-1}(\mathcal{Z}) = 1] - \mathsf{Pr}[\mathrm{out}_{2,i^*}(\mathcal{Z}) = 1]| > \epsilon/N$$

Moreover, if party P_{i^*} is not corrupted after input, i.e. if it is corrupted before input or remains honest throughout the execution, then the views of $\mathcal{Z}$ in H_{2,i^*-1} and H_{2,i^*} are identically distributed. Therefore,

$$\begin{aligned}\epsilon/N <& |\mathsf{Pr}[\mathrm{out}_{2,i^*-1}(\mathcal{Z}) = 1] - \mathsf{Pr}[\mathrm{out}_{2,i^*}(\mathcal{Z}) = 1]| \\ =& |\mathsf{Pr}[\mathrm{out}_{2,i^*-1}(\mathcal{Z}) = 1 \wedge \textbf{party } P_{i^*} \textbf{ corrupted after input}] \\ &- \mathsf{Pr}[\mathrm{out}_{2,i^*}(\mathcal{Z}) = 1 \wedge \textbf{party } P_{i^*} \textbf{ corrupted after input}]|\end{aligned}$$

Consider the following adversary $\mathcal{A}$ against the IND-pCCA security of PKE: At the beginning, $\mathcal{A}$ randomly selects an index $j \in \{1, \ldots, N\} \setminus \{i^*\}$. $\mathcal{A}$ then simulates the experiment H_{2,i^*-1}. When $\mathcal{Z}$ gives the party P_{i^*} its input x_{i^*}, $\mathcal{A}$ generates shares s_{i^*l}, r_{i^*l}, k_{i^*l} of the input x_{i^*}, of a random pad r_{i^*} and of a MAC key k_{i^*} just like in H_{2,i^*-1}. $\mathcal{A}$ additionally generates a random string k'_{i^*j}. $\mathcal{A}$ then generates signatures σ_{i^*j}, σ'_{i^*j} for $(P_j, s_{i^*j}, r_{i^*j}, k_{i^*j})$ and $(P_j, s_{i^*j}, r_{i^*j}, k'_{i^*j})$, respectively, and sends $(P_{i^*}, s_{i^*j}, r_{i^*j}, k_{i^*j}, \sigma_{i^*j})$, $(P_{i^*}, s_{i^*j}, r_{i^*j}, k'_{i^*j}, \sigma'_{i^*j})$ to the experiment, receiving a ciphertext c^*. Note that $\mathcal{A}$'s challenge messages are allowed, i.e. have the same length, because SIG is length-normal. $\mathcal{A}$ then continues simulating the experiment H_{2,i^*-1}, using c^* as c_{i^*j} and his decryption oracle to decrypt the ciphertexts in the buffer of P_j that are addressed as coming from the parties corrupted before input but do not equal c^* (the ones that are equal to c^* are ignored[29]). Note that in $\mathcal{A}$'s internal simulation, party P_{i^*} receives the correct value from $\mathcal{F}_\mathcal{G}$ (i.e. $(y_{i^*} + r_{i^*}, \mathrm{Mac}(k_{i^*}, y_{i^*} + r_{i^*}))$ or $\bot$). At the end of the experiment, $\mathcal{A}$ outputs what $\mathcal{Z}$ outputs. If during the simulation, $\mathcal{Z}$ corrupts P_j (before or after input) or if P_{i^*} is *not* corrupted after input, $\mathcal{A}$ sends $\bot$ to the experiment.

Let $\mathrm{Output}^{\mathsf{ind\text{-}pcca}}_{\mathrm{PKE},\mathcal{A}}(b)$ denote the output of $\mathcal{A}$ in the IND-pCCA experiment when the challenge bit b is chosen (cp. Section 3.1.1). By construction, assuming party P_{i^*} is corrupted after input, if $\mathcal{A}$ guessed an index j such that party P_j remains honest then it holds that if the challenge bit is 0 the view of $\mathcal{Z}$ in $\mathcal{A}$'s internal simulation is distributed as in the experiment H_{2,i^*-1} and if the challenge bit is 1 the view of $\mathcal{Z}$ in $\mathcal{A}$'s internal simulation is distributed as in the experiment H_{2,i^*}. Moreover, assuming party P_{i^*} is corrupted after input, the probability that $\mathcal{A}$ guesses an index j such that party P_j remains honest is at least $1/(N-1)$. Hence,

$$\begin{aligned}
&|\mathrm{Output}^{\mathsf{ind\text{-}pcca}}_{\mathrm{PKE},\mathcal{A}}(0) - \mathrm{Output}^{\mathsf{ind\text{-}pcca}}_{\mathrm{PKE},\mathcal{A}}(1)| \\
=&|\mathsf{Pr}[\mathrm{out}_{2,i^*-1}(\mathcal{Z}) = 1 \wedge \textbf{party } P_{i^*} \textbf{ corrupted after input} \wedge \textbf{Guess correct}] \\
&- \mathsf{Pr}[\mathrm{out}_{2,i^*}(\mathcal{Z}) = 1 \wedge \textbf{party } P_{i^*} \textbf{ corrupted after input} \wedge \textbf{Guess correct}]| \\
>& \epsilon/(N \cdot (N-1))
\end{aligned}$$

This contradicts the IND-pCCA security of PKE.

Hence, there exist a negligible function negl_2 such that

$$|\mathsf{Pr}[\mathrm{out}_1(\mathcal{Z}) = 1] - \mathsf{Pr}[\mathrm{out}_2(\mathcal{Z}) = 1]| \leq \mathsf{negl}_2(n)$$

Hybrid H_3. Let H_3 be the execution experiment between the environment $\mathcal{Z}$, the ideal protocol $\mathtt{AG}(\mathcal{F}_2)$ and the ideal-model adversary Sim_3, where $\mathcal{F}_2$ and Sim_3 are defined as follows:

Let $\mathcal{F}_2$ be identical to $\mathcal{F}_1$, except that now the adversary is *not* allowed to determine the outputs of the dummy OIMs of parties *corrupted after input* anymore (only of parties corrupted before input).

Define Sim_3 to be like Sim_2, except that Item 19 is identical to the same step of the simulator in Definition 5.15.

Let $\mathbf{E}_{\mathrm{fakeoutp}}$ be the event that $\mathcal{Z}$ sends a message (m', t') to OIM of a party P_i corrupted after input such that $\mathrm{Vrfy}_{\mathrm{MAC}}(k_i, m', t') = 1$ but either P_i has received $\bot$ from $\mathcal{F}_\mathcal{G}$ or a tuple (m, t) such that $m' \neq m$, or P_i has not received an output from $\mathcal{F}_\mathcal{G}$ yet.

It is easy to see that the following holds:

$$\mathsf{Pr}[\mathrm{out}_2(\mathcal{Z}) = 1 \wedge \neg\mathbf{E}_{\mathrm{fakeoutp}}] = \mathsf{Pr}[\mathrm{out}_3(\mathcal{Z}) = 1 \wedge \neg\mathbf{E}_{\mathrm{fakeoutp}}]$$

Therefore, it holds that

$$|\mathsf{Pr}[\mathrm{out}_2(\mathcal{Z}) = 1] - \mathsf{Pr}[\mathrm{out}_3(\mathcal{Z}) = 1]| \leq \mathsf{Pr}[\mathbf{E}_{\mathrm{fakeoutp}}]$$

[29]Note that a tuple (P_l, c^*) addressed as coming from a party P_l corrupted before input is always invalid since $P_l \neq P_{i^*}$.

Claim 3: $\mathsf{Pr}[\mathbf{E}_{\text{fakeoutp}}]$ is negligible.
Consider the adversary $\mathcal{A}$ against the EUF-1-CMA-security of MAC. At the beginning, $\mathcal{A}$ randomly selects an index $i \in \{1, \dots, N\}$. $\mathcal{A}$ then simulates the hybrid H_2. Once $\mathcal{Z}$ expects the output from $\mathcal{F}_{\mathcal{G}}$ for party P_i (if P_i is corrupted after input), $\mathcal{A}$ computes the output m for this party. If $m = \bot$, $\mathcal{A}$ sends $\bot$ to $\mathcal{Z}$. Otherwise, $\mathcal{A}$ sends m to the oracle $\mathcal{O}_{\mathrm{Mac}(k,\cdot)}$, receiving a tag t. $\mathcal{A}$ then sends (m, t) to $\mathcal{Z}$. If $\mathcal{Z}$ sends a tuple (m', t') to OIM of P_i such that $m' \neq m$, then $\mathcal{A}$ sends (m', t') to the experiment. If during the simulation, P_i is *not* corrupted after input or if $\mathcal{Z}$ sends $\bot$ or a tuple (m', t') such that $m' = m$ to OIM of P_i, then $\mathcal{A}$ sends $\bot$ to the experiment.

By construction, it holds that if $\mathbf{E}_{\text{fakeoutp}}$ occurs and $\mathcal{A}$ correctly guessed an index for which $\mathbf{E}_{\text{fakeoutp}}$ occurs, then $\mathsf{Exp}^{\text{euf-1-cma}}_{\mathrm{MAC},\mathcal{A}(z)}(n) = 1$ because (m', t') is valid and $m' \neq m$ has not been sent to $\mathcal{O}_{\mathrm{Mac}(k,\cdot)}$. Moreover, the probability that $\mathcal{A}$ correctly guesses an index for which $\mathbf{E}_{\text{fakeoutp}}$ occurs is at least $1/N$. Hence,

$$\mathsf{Pr}[\mathsf{Exp}^{\text{euf-1-cma}}_{\mathrm{MAC},\mathcal{A}(z)}(n) = 1] \geq \mathsf{Pr}[\mathbf{E}_{\text{fakeoutp}}]/N$$

Therefore, since $\mathsf{Pr}[\mathsf{Exp}^{\text{euf-1-cma}}_{\mathrm{MAC},\mathcal{A}(z)}(n) = 1]$ is negligible because MAC is EUF-1-CMA-secure by assumption and N is polynomial in n, it follows that $\mathsf{Pr}[\mathbf{E}_{\text{fakeoutp}}]$ is also negligible.

Hence, there exist a negligible function negl_3 such that

$$|\mathsf{Pr}[\mathrm{out}_2(\mathcal{Z}) = 1] - \mathsf{Pr}[\mathrm{out}_3(\mathcal{Z}) = 1]| \leq \mathsf{negl}_3(n)$$

Hybrid H_4. Let H_4 be the execution experiment between $\mathcal{Z}$, the ideal protocol $\mathtt{AG}(\mathcal{F}_3)$ and the ideal-model adversary Sim_4, where $\mathcal{F}_3$ and Sim_4 are defined as follows: Let $\mathcal{F}_3$ be identical to $\mathcal{F}_2$, except that the adversary is *not* given the inputs and outputs of honest parties anymore. In addition, the adversary is only given the inputs and outputs of parties corrupted after input when all parties are corrupted.

Define the adversary Sim_4 to be like Sim_3, except that Items 8, 10 and 18 are identical to the same steps of the simulator in Definition 5.15.

Let $\mathrm{H}_{4,0}, \dots, \mathrm{H}_{4,N}$ be the execution experiment between the environment $\mathcal{Z}$, the ideal protocol $\mathtt{AG}(\mathcal{F}_{3,0}), \dots, \mathtt{AG}(\mathcal{F}_{3,N})$ and the adversary $\mathsf{Sim}_{4,0}, \dots, \mathsf{Sim}_{4,N}$, respectively, where $\mathcal{F}_{3,i}$ and $\mathsf{Sim}_{4,i}$ are defined as follows:

Define $\mathcal{F}_{3,i}$ be identical to $\mathcal{F}_2$, except now the adversary is not given the inputs and outputs of the parties $P_l \in \{P_1, \dots, P_i\}$ if they are honest or corrupted after input unless all parties are corrupted.

Define the simulators $\mathsf{Sim}_{4,i}$ to be like Sim_3, except for the following: In Item 8, upon notification by $\mathcal{F}_{3,i}$ that an (*honest*) party $P_l \in \{P_1, \dots, P_i\}$ has sent its input, $\mathsf{Sim}_{4,i}$ generates N random strings $s'_{l1}, \dots, s'_{lN}$ of length p, N random strings $r'_{l1}, \dots, r'_{lN}$ of length q and N random strings $k'_{l1}, \dots, k'_{lN}$ of length $|k_l|$ (where k_l is the MAC key generated in Item 6). $\mathsf{Sim}_{4,i}$ then computes $\sigma'_{lj} \leftarrow \mathrm{Sig}(\mathsf{sgk}_l, j, s'_{lj}, r'_{lj}, k'_{lj})$ $(j = 1, \dots, N)$, and iteratively reports $(P_l, \mathrm{Enc}(\mathsf{pk}_j, P_l, s'_{lj}, r'_{lj}, k'_{lj}, \sigma'_{lj}))$ $(j \in \{1, \dots, N\} \setminus \{l\})$ to $\mathcal{Z}$. If a party $P_l \in \{P_1, \dots, P_i\}$ is corrupted after having received input, $\mathsf{Sim}_{4,i}$ sends $(s'_{ll}, r'_{ll}, k'_{ll}, \sigma'_{ll}, \mathsf{vk}_l, \mathsf{sk}_l)$ to $\mathcal{Z}$ in Item 10. In Item 18, if $\mathtt{verify} = \mathtt{true}$, then for every party $P_l \in \{P_1, \dots, P_i\}$ corrupted after having received input, Sim_4 generates a random string $\tilde{y}_l \leftarrow \{0,1\}^q$ and sends $(\tilde{y}_l, \mathrm{Mac}(k_l, \tilde{y}_l))$ to $\mathcal{Z}$ as the output from $\mathcal{F}_{\mathcal{G}}$ (where k_l is the MAC key generated in Item 6). If `verify` $=$ `false`, then for every corrupted party, $\mathsf{Sim}_{4,i}$ sends $\perp$ to $\mathcal{Z}$ as the output from $\mathcal{F}_{\mathcal{G}}$.

It holds that

$$\mathsf{Pr}[\mathrm{out}_{4,0}(\mathcal{Z}) = 1] = \mathsf{Pr}[\mathrm{out}_3(\mathcal{Z}) = 1]$$

and

$$\mathsf{Pr}[\mathrm{out}_{4,N}(\mathcal{Z}) = 1] = \mathsf{Pr}[\mathrm{out}_4(\mathcal{Z}) = 1]$$

Assume that there exists a non-negligible function ϵ such that $|\mathsf{Pr}[\mathrm{out}_3(\mathcal{Z}) = 1] = \mathsf{Pr}[\mathrm{out}_4(\mathcal{Z}) = 1]| > \epsilon$. Then there exists an $i^* \in \{1, \ldots, N\}$ such that

$$|\mathsf{Pr}[\mathrm{out}_{4,i^*-1}(\mathcal{Z}) = 1] = \mathsf{Pr}[\mathrm{out}_{4,i^*}(\mathcal{Z}) = 1]| > \epsilon/N$$

Moreover, if party P_{i^*} is not corrupted after input, i.e. if it is corrupted before input or remains honest throughout the execution, then the views of $\mathcal{Z}$ in H_{4,i^*-1} and H_{4,i^*} are identically distributed. Therefore,

$$\begin{aligned}\epsilon/N &< |\mathsf{Pr}[\mathrm{out}_{4,i^*-1}(\mathcal{Z}) = 1] - \mathsf{Pr}[\mathrm{out}_{4,i^*}(\mathcal{Z}) = 1]| \\ &= |\mathsf{Pr}[\mathrm{out}_{4,i^*-1}(\mathcal{Z}) = 1 \wedge \textbf{party } P_{i^*} \textbf{ corrupted after input}] \\ &\quad - \mathsf{Pr}[\mathrm{out}_{4,i^*}(\mathcal{Z}) = 1 \wedge \textbf{party } P_{i^*} \textbf{ corrupted after input}]|\end{aligned}$$

Consider the following adversary $\mathcal{A}$ against the IND-pCCA security of PKE: At the beginning, $\mathcal{A}$ randomly selects an index $j \in \{1, \ldots, N\} \setminus \{i^*\}$. $\mathcal{A}$ then simulates the experiment H_{4,i^*-1}. When $\mathcal{Z}$ gives the party P_{i^*} its input x_{i^*}, $\mathcal{A}$ generates shares s_{i^*l} and r_{i^*l} of x_{i^*} and of a random pad r_{i^*} and generates random strings k'_{i^*l} ($l \in \{1, \ldots, N\}$) just like in H_{4,i^*-1}. $\mathcal{A}$ additionally generates random strings s'_{i^*j} and r'_{i^*j}. $\mathcal{A}$ then generates signatures $\sigma_{i^*j}, \sigma'_{i^*j}$ for $(P_j, s_{i^*j}, r_{i^*j}, k'_{i^*j})$ and $(P_j, s'_{i^*j}, r'_{i^*j}, k'_{i^*j})$, respectively, and sends $(P_{i^*}, s_{i^*j}, r_{i^*j}, k'_{i^*j}, \sigma_{i^*j})$, $(P_{i^*}, s'_{i^*j}, r'_{i^*j}, k'_{i^*j}, \sigma'_{i^*j})$ to the experiment, receiving a ciphertext c^*. Note that $\mathcal{A}$'s challenge messages are allowed because SIG is length-normal. $\mathcal{A}$ then continues simulating the experiment H_{4,i^*-1}, using c^* as c_{i^*j} and his decryption oracle to decrypt the ciphertexts in the buffer of P_j that are addressed as coming from the parties corrupted before input but do not equal c^* (the ones that are equal to c^* are ignored, cf. Footnote 29). Note that in $\mathcal{A}$'s internal simulation, party P_{i^*} receives the correct value from $\mathcal{F}_{\mathcal{G}}$ (i.e. $(y_{i^*} + r_{i^*}, \mathrm{Mac}(k_{i^*}, y_{i^*} + r_{i^*})$ or $\perp$). At the end of the experiment, $\mathcal{A}$ outputs what $\mathcal{Z}$ outputs. If during the simulation, $\mathcal{Z}$ corrupts P_j (before or after input) or if party P_{i^*} is *not* corrupted after input, then $\mathcal{A}$ sends $\perp$ to the experiment.

By construction, assuming party P_{i^*} is corrupted after input, if $\mathcal{A}$ guessed an index j such that party P_j remains honest then it holds that if the challenge bit is 0 the view of $\mathcal{Z}$ in $\mathcal{A}$'s internal simulation is distributed as in the experiment H_{4,i^*-1} and if the challenge bit is 1 the view of $\mathcal{Z}$ in $\mathcal{A}$'s internal simulation is distributed as in the experiment H_{4,i^*}. Moreover, assuming party P_{i^*} is corrupted after input, the probability that $\mathcal{A}$ guesses an index j such that party P_j remains honest is at least $1/(N-1)$. Hence,

$$\begin{aligned}&|\mathrm{Output}^{\textsf{ind-pcca}}_{\mathrm{PKE},\mathcal{A}}(0) - \mathrm{Output}^{\textsf{ind-pcca}}_{\mathrm{PKE},\mathcal{A}}(1)| \\ =&|\mathsf{Pr}[\mathrm{out}_{4,i^*-1}(\mathcal{Z}) = 1 \wedge \textbf{party } P_{i^*} \textbf{ corrupted after input} \wedge \textbf{Guess correct}] \\ &- \mathsf{Pr}[\mathrm{out}_{4,i^*}(\mathcal{Z}) = 1 \wedge \textbf{party } P_{i^*} \textbf{ corrupted after input} \wedge \textbf{Guess correct}]| \\ &> \epsilon/(N \cdot (N-1))\end{aligned}$$

This contradicts the IND-pCCA security of PKE.

Hence, there exists a negligible function negl_3 such that

$$|\Pr[\text{out}_3(\mathcal{Z}) = 1] - \Pr[\text{out}_4(\mathcal{Z}) = 1]| \leq \mathsf{negl}_3(n)$$

Since H_4 is identical to the execution between $\mathcal{Z}$, the ideal protocol $\mathtt{AG}([\mathcal{G}])$ and the simulator Sim defined in Definition 5.15, it follows that there exists a negligible function negl such that

$$|\Pr[\text{Exec}_{\text{FortUC}}(\Pi_{\mathcal{G}}, \mathcal{D}, \mathcal{Z}) = 1] - \Pr[\text{Exec}_{\text{FortUC}}(\mathtt{AG}([\mathcal{G}]), \mathsf{Sim}, \mathcal{Z}) = 1]| \leq \mathsf{negl}(n)$$

The statement follows. □

Remark 5.17. *Using Theorems 5.9 and 5.10, we can replace $\overline{\mathtt{SC}(\mathcal{F}_{\mathcal{G}})}$ in $\Pi_{\mathcal{G}}$ with an adaptively UC-secure protocol (based on standard polynomial-time assumptions), e.g. [Can+02; Dac+13b; HV15]. Note that this inevitably requires an additional trusted setup assumption (e.g. a common reference string, cf. Definition 2.24) because the unhackable sub-parties (and $\mathcal{F}_{\mathsf{reg}}$) cannot be used to circumvent the impossibility results in the UC framework (e.g. [CF01; CKL03]).*

Remark 5.18. *PKE schemes, digital signature schemes and MACs satisfying the respective properties in Theorem 5.16 exist under the assumption that trapdoor permutations exist. More specifically, IND-CPA-secure PKE schemes exist assuming the existence of trapdoor permutations [Gol04]. [PSV06; Cho+18] showed how to construct a NM-CPA-secure PKE scheme given any IND-CPA secure PKE scheme without further assumptions. Furthermore, lenght-normal EUF-naCMA-secure digital signatures schemes can be constructed assuming the existence of one-way functions [Rom90; Gol04]. Finally, EUF-1-CMA-secure MACs exist if one-way functions exist [GGM84; GGM86].*

Remark 5.19. *Note that one can also let a party check each message it receives (in its buffer) right away once it is online without having to wait for at least $N-1$ messages in the buffer. The protocol remains secure if one assumes the stronger assumption that* PKE *is IND-CCA-secure (cf. Definition 3.1 for a definition).*

Remark 5.20. *Note that if the parties P_i disconnect all their air-gap switches again after receiving output from $\mathcal{F}_{\mathcal{G}}$, then $\mathcal{A}$ cannot corrupt them anymore and thus cannot obtain all shares anymore.*

5.4.1 Up to N Parties Under Adversarial Control

One can augment construction 4 in order to obtain a protocol $\Pi_{\mathcal{G}}^{(2)}$ that is also secure if the adversary corrupts *all* parties at the expense of one additional unhackable sub-party called *decryption unit* (Dec-unit). The main idea in the new construction is that the parties do not decrypt ciphertexts themselves but instead send them to their Dec-unit (cf. Fig. 5.5). More specifically, in the sharing phase, each party P_i sends its decryption key sk_i to its Dec-unit via a data diode and then deletes sk_i. In the compute phase, each Dec-unit accepts a single vector of ciphertexts from its main party (via a connected air-gap switch).

The simulator Sim' for the case of up to N parties under adversarial control is identical to the simulator for up to $N-1$ in Definition 5.15, except for the following: Once the Nth (final) party has been corrupted, Sim', who learns the

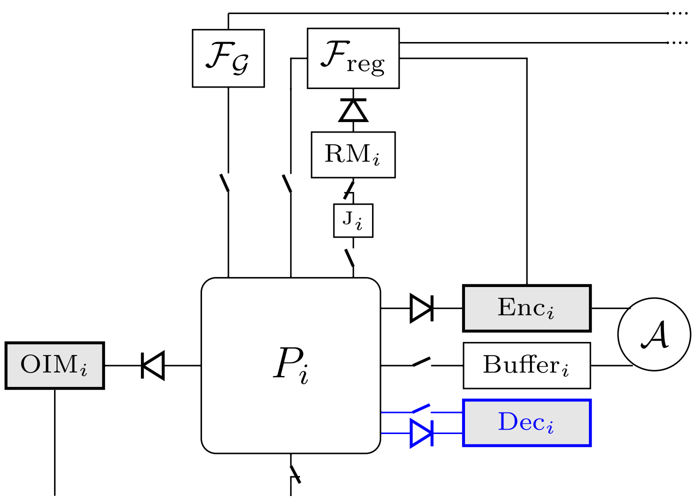


Figure 5.5: Architecture of $\Pi_{\mathcal{G}}^{(2)}$: Each party P_i ($i = 1, \dots, N$) has 3 hackable sub-parties, called *buffer*, *registration module* (RM) and *join* (J), and 3 unhackable sub-parties, called Enc(-unit), Dec(-unit) and OIM. Buffer and Enc-unit are connected to the adversary via standard channels. All air-gap switches, except for P's air-gap switch to the environment and the RM's air-gap switch to J, are initially *disconnected.*

inputs and outputs of all parties from $[\mathcal{G}]$ in this case, reports a plaintext tuple such that the shares it contains and the shares received by the other parties are consistent with the input, MAC key and output of the Nth corrupted party. If $\mathcal{Z}$ sends a vector of ciphertexts to the Dec-unit of the Nth (final) party that has been corrupted, Sim' returns plaintext tuples such that the shares they contain and the shares held by the other parties are consistent with the inputs, MAC keys and outputs of the other parties. Note that $\mathcal{Z}$ is unable to check if the plaintext tuples it receives from Sim' have been encrypted before since the Dec-units are unhackable and do not leak the decryption keys.

More specifically, let $s'_{ij}, r'_{ij}, k'_{ij}$ be the $3N$ shares which are generated in Item 8 and $\tilde{y}_i \leftarrow \{0,1\}^n$ ($i = 1, \dots, N$) the random strings which are generated in Item 18. Once the Nth (final) party, denoted by P_{l^*}, is corrupted, Sim' computes for each i the values $\tilde{s}_{il^*} = x_i + \sum_{j \in \{1,\dots,N\} \setminus \{l^*\}} s'_{ij}$, and $\tilde{k}_{il^*} = k_i + \sum_{j \in \{1,\dots,N\} \setminus \{l^*\}} k'_{ij}$ and $\tilde{r}_{il^*} = \tilde{y}_i + y_i + \sum_{j \in \{1,\dots,N\} \setminus \{l^*\}} r'_{ij}$. Sim' then creates signatures $\tilde{\sigma}_{il^*} \leftarrow \mathrm{Sig}(\mathsf{sgk}_i, P_{l^*}, \tilde{s}_{il^*}, \tilde{r}_{il^*}, \tilde{k}_{il^*})$ ($i = 1, \dots, N$) (where the sgk_i are the signing keys generated in Item 6) and reports the plaintext tuple $(P_i, \tilde{s}_{l^*l^*}, \tilde{r}_{l^*l^*}, \tilde{k}_{l^*l^*}, \tilde{\sigma}_{l^*l^*})$ to $\mathcal{Z}$. If $\mathcal{Z}$ sends a vector of ciphertexts to the Dec-unit of party P_{l^*}, then Sim' checks for each c' contained in that vector if $c' = c_{il^*}$ for some i. For each c' for which this holds, Sim' returns the corresponding plaintext tuple $(P_i, \tilde{s}_{il^*}, \tilde{r}_{il^*}, \tilde{k}_{il^*}, \tilde{\sigma}_{il^*})$. For each c' for which this does not hold, Sim' returns $\mathrm{Dec}(\mathsf{sk}_l^*, c')$ (where the sk_i are the decryption keys generated in Item 6). Note that for each i it holds that $x_i = \tilde{s}_{il^*} + \sum_{j \in \{1,\dots,N\} \setminus \{l^*\}} s'_{ij}$ and $k_i = \tilde{k}_{il^*} + \sum_{j \in \{1,\dots,N\} \setminus \{l^*\}} k'_{ij}$ and $y_i = \tilde{y}_i + \tilde{r}_{il^*} + \sum_{j \in \{1,\dots,N\} \setminus \{l^*\}} r'_{ij}$.

The proof that $\Pi_{\mathcal{G}}^{(2)}$ realizes $\mathtt{AG}([\mathcal{G}])$ for up to N parties under adversarial control is very similar to the proof of Theorem 5.16 and therefore omitted.

Theorem 5.21 (Up to N Corrupted Parties, Non-Reactive Case)**.** *Let $\mathcal{G}$ be a* non-reactive *standard adaptively well-formed functionality. Assume* PKE, SIG, MAC *are as in Theorem 5.16. Then it holds that*

$$\Pi_{\mathcal{G}}^{(2)} \underset{\#\#}{\geq} \mathtt{AG}([\mathcal{G}])$$

for up to N parties under adversarial control.

5.5 Construction for Reactive Functionalities

In this section, we present a construction for realizing the fortified functionality of every *reactive* (standard adaptively well-formed) ideal functionality. The new construction is a direct generalization of constructions 3 and 4.

For reactive functionalities, a new problem arises because a protocol party is online after the first round. The input(s) for the next round(s) can therefore not just be given to a party since it may have been corrupted. We therefore need to find a way to insert the input(s) for the rounds $u \geq 2$ into the protocol without allowing a party to learn or modify them.

To this end, we introduce an additional unhackable sub-party called *input interface module* (IIM) which acts as the counterpart of the OIM for inputs. Let $\mathrm{R} \in \mathbb{N}$ be the number of rounds. In the sharing phase, each party P_i generates $2\mathrm{R}$ random pads $r_i^1, \ldots, r_i^{\mathrm{R}}, t_i^1, \ldots, t_i^{\mathrm{R}}$ and creates shares of these random pads. Also, each party P_i pads its (first) input $\tilde{x}_i^1 := x_i^1 + t_i^1$ and computes a MAC tag of this padded input. Each party P_i then sends the R random pads $r_i^1, \ldots, r_i^{\mathrm{R}}$ as well as the MAC key k_i to its OIM and the other R random pads $t_i^1, \ldots, t_i^{\mathrm{R}}$ and the MAC key k_i to its IIM. As before, each random pad is shared with the other parties along with signatures of these shares, the PID of the designated receiver as well as the *number of the round* in which this share is to be used. Note that the latter prevents an adversary from re-using shares from earlier rounds.

In each compute phase, the parties use their shares, signatures, padded inputs and MAC tags to compute the desired padded output values for that round as well as MAC tags of these padded output values and of a *prefix containing the round number and an indication that this value is an output.* Verification and reconstruction of the output values is then done as before using the OIM. Note that since the prefix contains the round number, an OIM is able to reject results from earlier compute phases.

As before, each input to the compute phase has to be verified before the desired padded output values are computed. Now, however, not only the signatures of the shares are verified but also the MAC tags of the padded inputs. In order to obtain the MAC tags for the padded inputs for the rounds $u \geq 2$, the respective input has to be inserted into the protocol via the IIM. The IIM applies a one-time pad on each input it receives and computes a MAC tag of the padded input and of a *prefix containing the round number and an indication that this value is an input.* It then sends the computed tuple to the party. This way, a party is be able to continue the computation without learning the inputs for the rounds $u \geq 2$. Note that since the prefix contains the round number, the adversary cannot use padded inputs of *earlier* rounds. Also note that since the prefix indicates whether the respective value is an input or output, an adversary cannot send a padded input to an OIM.

As before, we will take a modular approach and define an ideal functionality $\mathcal{F}_{\mathcal{G}}^{\mathtt{reac}}$ that implements the verification of the input values in the compute phase as well as the multi-party computation on the shares and padded inputs. $\mathcal{F}_{\mathcal{G}}^{\mathtt{reac}}$ is defined as follows.

Construction 5.
Let $\mathcal{G}$ be a reactive standard adaptively well-formed ideal functionality with R *rounds. $\mathcal{F}_{\mathcal{G}}^{\mathtt{reac}}$ proceeds as follows, running with parties $P_1, \ldots, P_N$ and an adversary $\mathcal{A}$ and parametrized with a digital signature* SIG *and a message authentication code* MAC*.*

1. *Initialize* $\mathrm{R}+1$ *Boolean variables* $\mathtt{verify}^0$, $\mathtt{verify}^1, \ldots, \mathtt{verify}^{\mathrm{R}} = \mathtt{true}$ *and a counter* $u = 1$*.*
2. *Upon receiving input from party P_i, store it and send* $(\mathtt{received}, P_i)$ *to* $\mathcal{A}$*. Upon receiving* $(\mathtt{confirmed}, P_i)$ *from* $\mathcal{A}$*, mark P_i as* input given*.*
3. *Upon receiving* $(\mathtt{corrupt}, P_i)$*, behave like a standard corruption ideal functionality. In addition, forward this message to* $\mathcal{G}$*.*
4. *Upon receiving from $\mathcal{A}$ a (modified) input for a party P_l marked as* corrupted*, store that input (if an input has already been stored for P_l then overwrite it) and, if not done yet, mark P_l as* input given*.*

Consistency Check

5. *Once each party has been marked as* input given*, delete all* input given *markings and proceed with Item 6 if this is round $u = 1$, else proceed with Item 7.*
6. *Check if every party P_i has sent an input of the form* $\overrightarrow{\mathsf{vk}}_i = (\mathsf{vk}_1^{(i)}, \ldots, \mathsf{vk}_N^{(i)})$, $(t_{ji}, r_{ji}, \sigma_{ji}, k_{ji}, \sigma'_{ji})$ $(j = 1, \ldots, N)$*.*
   i) *If no, set* $\mathtt{verify}^0 = \mathtt{false}$*.*
   ii) *If yes, check if* $\overrightarrow{\mathsf{vk}}_1 = \cdots = \overrightarrow{\mathsf{vk}}_N$*.*
      (A) *If this does not hold, set* $\mathtt{verify} = \mathtt{false}$*.*
      (B) *Else, set* $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n) = (\mathsf{vk}_1^{(1)}, \ldots, \mathsf{vk}_N^{(1)})$*. For all* $i = 1, \ldots, N$*, check if* $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathsf{vk}_j, P_i, k_{ji}, \sigma'_{ji}) = 1$ *for all* $j = 1, \ldots, N$*.*
         (a) *If this does not hold, set* $\mathtt{verify} = \mathtt{false}$*.*
         (b) *Else, for each* $i = 1, \ldots, N$*, compute and store* $k_i = k_{i1} + k_{i2} + \cdots + k_{iN}$ *and continue with Item 8.*
7. *If* $\mathtt{verify}^0 = \mathtt{false}$*, do nothing. Else, check if every party P_i has sent an input of the form* $(t_{ji}^u, r_{ji}^u, \sigma_{ji}^u)$ $(j = 1, \ldots, N)$, $(\tilde{x}_i^u, \tau_i^u)$*. If no, set* $\mathtt{verify}^{\mathtt{u}} = \mathtt{false}$*. Else, continue with Item 8.*
8. *For all* $i = 1, \ldots, N$*, check if* $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathsf{vk}_j, u, P_i, t_{ji}^u, r_{ji}^u, \sigma_{ji}^u) = 1$ *for all* $j = 1, \ldots, N$ *and if* $\mathrm{Vrfy}_{\mathrm{MAC}}(k_i, \mathtt{Inp\ Round\ u}, \tilde{x}_i^u, \tau_i^u) = 1$*.*
   (a) *If this does not hold for all i, j, set* $\mathtt{verify}^{\mathtt{u}} = \mathtt{false}$*.*
   (b) *Else, proceed with Item 9.*

Reconstruction and Computation

9. *For each $i = 1, \dots, N$, compute $r_i^u = r_{i1}^u + r_{i2}^u + \cdots + r_{iN}^u$ and $t_i^u = t_{i1}^u + t_{i2}^u + \cdots + t_{iN}^u$ and $x_i^u = \tilde{x}_i^u + t_i^u$.*

10. *Internally run $\mathcal{G}$ on input $(x_i^u, \dots, x_N^u)$. Let $(y_1^u, \dots, y_N^u)$ be the output of $\mathcal{G}$. For all $i = 1, \dots, N$, compute $o_i^u = y_i^u + r_i^u$ and $\theta_i^u \leftarrow \mathrm{Mac}(k_i, \texttt{Outp Round u}, o_i^u)$. Increment counter u.*

11. *If party P_i requests an output for round u', proceed as follows:*

    (i) If $u \leq u'$, ignore.

    (ii) Else, if $\texttt{verify}^0 = \texttt{false}$ or $\texttt{verify}^{\texttt{u}'} = \texttt{false}$, send a private delayed output $\perp$ to P_i.

    (iii) Else, send a private delayed output (o_i^u, θ_i^u) to P_i.

12. *Once all parties are corrupted, send all of its private randomness used so far as well as the private randomness $\mathcal{G}$ sends to $\mathcal{A}$ in this case (note that $\mathcal{G}$ is adaptively well-formed) to the adversary $\mathcal{A}$. (Note that this ensures that $\mathcal{F}_{\mathcal{G}}^{\texttt{reac}}$ is also adaptively well-formed).*

Let $\mathcal{G}$ be a *reactive* standard adaptively well-formed functionality. We next define our protocol $\Pi_{\mathcal{G}}^{(3)}$ for realizing the ideal protocol $\texttt{AG}([\mathcal{G}])$.

Construction 6. *Define the protocol $\Pi_{\mathcal{G}}^{(3)}$ as follows:*
Architecture: *See Fig. 5.6 for a graphical depiction.*

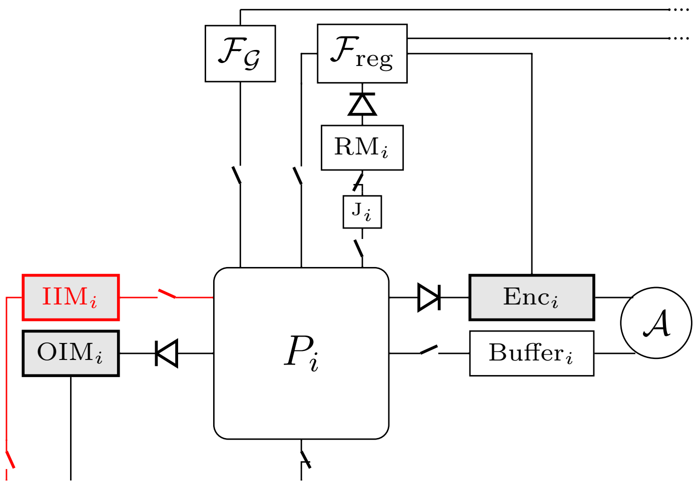


Figure 5.6: Architecture of $\Pi_{\mathcal{G}}^{(3)}$: Each party P_i $(i = 1, \dots, N)$ has 3 hackable sub-parties, called *buffer*, *registration module* (RM) and *join* (J), and 3 unhackable sub-parties, called Enc(-unit), OIM and IIM. Buffer and Enc-unit are connected to the adversary via standard channels. All air-gap switches, except for P's air-gap switch to the environment and the RM's air-gap switch to J, are initially *disconnected.*

Offline Sharing Phase
Upon input $x_i^1 \in \{0,1\}^p$, each party P_i does the following:

- Disconnect *air-gap switch to the environment.*
- *Generate a key pair* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathrm{Gen}_{\mathrm{PKE}}(1^n)$*, a MAC key* $k_i \leftarrow \mathrm{Gen}_{\mathrm{MAC}}(1^n)$*, a signature key pair* $(\mathsf{sgk}_i, \mathsf{vk}_i) \leftarrow \mathrm{Gen}_{\mathrm{SIG}}(1^n)$ *and* $2\mathrm{R}$ *random pads* $t_i^1, t_i^2, \ldots, t_i^{\mathrm{R}} \leftarrow \{0,1\}^p$ *and* $r_i^1, r_i^2, \ldots, r_i^{\mathrm{R}} \leftarrow \{0,1\}^q$.
- *Generate shares* $t_{i1}^u + t_{i2}^u + \cdots + t_{iN}^u = t_i^u$ *and* $r_{i1}^u + r_{i2}^u + \cdots + r_{iN}^u = r_i^u$ $(u = 1, \ldots, \mathrm{R})$ *and* $k_{i1} + k_{i2} + \cdots + k_{iN} = k_i$.
- *Connect air-gap switch to J and to* IIM.
- *Send* (k_i, r_i^u) *to the* OIM *and* (k_i, t_i^u) *to the* IIM $(u = 1, \ldots, \mathrm{R})$.
- *Send* $(\mathsf{pk}_i, \mathsf{vk}_i)$ *to the registration module via J and to* IIM.
- *Create signatures* $\sigma_{ij}^u \leftarrow \mathrm{Sig}(\mathsf{sgk}_i, u, P_j, t_{ij}^u, r_{ij}^u)$ *and* $\sigma'_{ij} \leftarrow \mathrm{Sig}(\mathsf{sgk}_i, P_j, k_{ij})$ $(j = 1, \ldots, N; u = 1, \ldots, \mathrm{R})$.
- *Compute* $\tilde{x}_i^1 = x_i^1 + t_i^1$ *and* $\tau_i^1 \leftarrow \mathrm{Mac}(k_i, \texttt{Inp Round 1}, \tilde{x}_i^1)$
- *Let* $\vec{t}_{ij} = (t_{ij}^1, t_{ij}^2, \ldots, t_{ij}^{\mathrm{R}})$, $\vec{r}_{ij} = (r_{ij}^1, r_{ij}^2, \ldots, r_{ij}^{\mathrm{R}})$ *and* $\vec{\sigma}_{ij} = (\sigma_{ij}^1, \sigma_{ij}^2, \ldots, \sigma_{ij}^{\mathrm{R}})$. *Send* $(j, \vec{t}_{ij}, \vec{r}_{ij}, \vec{\sigma}_{ij}, k_{ij}, \sigma'_{ij})$ $(j \in \{1, \ldots, \mathrm{R}\} \setminus \{i\})$ *to the* Enc*-unit*
- Erase *everything, except for the tuple* $(\vec{t}_{ii}, \vec{r}_{ii}, \vec{\sigma}_{ii}, k_{ii}, \sigma'_{ii})$ *and* $(\tilde{x}_i^1, \tau_i^1)$ *and* $\mathsf{vk}_i, \mathsf{sk}_i$.

Registration module and J: *On input* $(\mathsf{pk}_i, \mathsf{vk}_i)$ *to J, J forwards the input to RM. RM then* disconnects *air-gap switch to J and registers* pk_i *and* vk_i *by sending these keys to the public bulletin board functionality* $\mathcal{F}_{\mathsf{reg}}$.

Enc-unit: *Receive a list* $L = \{(P_j, v_j)\}_{j=\{1,\ldots,N\}\setminus\{i\}}$ *from one's main party* P_i. *At each activation, for each* $(P_j, v_j) \in L$*, request* pk_j *belonging to* P_j *from* $\mathcal{F}_{\mathsf{reg}}$. *If retrievable, compute* $c_{ij} \leftarrow \mathrm{Enc}(\mathsf{pk}_j, v_j)$*, send* (P_i, c_{ij}) *to the buffer of* P_j *and delete* (P_j, v) *from L. Then, go into idle mode.*

Buffer: *Store each message received. On input* `retrieve`*, send all stored messages to one's main party.*

First Online Compute Phase
Having completed its last step in the sharing phase, each party P_i *does the following:*

- Connect *air-gap switches to the* $buffer$*, to* $\mathcal{F}_{\mathsf{reg}}$ *and to* $\mathcal{F}_{\mathcal{G}}^{\mathtt{reac}}$.
- *Request all verification keys* $\{\mathsf{vk}_l\}_{l\in\{1,\ldots,N\}\setminus\{i\}}$ *from* $\mathcal{F}_{\mathsf{reg}}$ *registered by the other parties'* registration modules. *If not all verification keys can be retrieved yet, go into idle mode and request again at the next activation.*

- *Send* `retrieve` *to the buffer and check if the buffer sends at least $N-1$ messages. If no, go into idle mode and when activated again send* `retrieve` *and check again.*
  If yes, check if one has received from each party P_j a set $\mathcal{M}_j = \{(P_j, \tilde{c})\}$ with the following property:
  There exists a tuple $(P_j, \hat{\vec{t}}_{ji}, \hat{\vec{r}}_{ji}, \hat{\vec{\sigma}}_{ji}, \hat{k}_{ji}, \hat{\sigma}'_{ji})$, where $\hat{\vec{t}}_{ji} = (\hat{t}^1_{ji}, \hat{t}^2_{ji}, \dots, \hat{t}^{\mathrm{R}}_{ji})$, $\hat{\vec{r}}_{ji} = (\hat{r}^1_{ji}, \hat{r}^2_{ji}, \dots, \hat{r}^{\mathrm{R}}_{ji})$ and $\hat{\vec{\sigma}}_{ji} = (\hat{\sigma}^1_{ji}, \hat{\sigma}^2_{ji}, \dots, \hat{\sigma}^{\mathrm{R}}_{ji})$, and a $(P_j, c) \in \mathcal{M}_j$ such that

  - $\mathrm{Dec}(\mathsf{sk}_i, c) = (P_j, \hat{\vec{t}}_{ji}, \hat{\vec{r}}_{ji}, \hat{\vec{\sigma}}_{ji}, \hat{k}_{ji}, \hat{\sigma}'_{ji})$ *and* $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathsf{vk}_j, u, P_i, \hat{t}^u_{ji}, \hat{r}^u_{ji}, \hat{\sigma}^u_{ji}) = 1$ $(u = 1, \dots, \mathrm{R})$ *and* $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathsf{vk}_j, P_i, \hat{k}_{ji}, \hat{\sigma}'_{ji}) = 1$
  - *For every $(P_j, \tilde{c}) \in \mathcal{M}_j$ it holds that either* $\mathrm{Dec}(\mathsf{sk}_i, \tilde{c}) = (P_j, \hat{\vec{t}}_{ji}, \hat{\vec{r}}_{ji}, \hat{\vec{\sigma}}_{ji}, \hat{k}_{ji}, \hat{\sigma}'_{ji})$ *or $(P_j, \tilde{c})$ is "invalid", i.e., either decrypts to a tuple $(P_j, \tilde{\vec{t}}_{ji}, \tilde{\vec{r}}_{ji}, \tilde{\vec{\sigma}}_{ji}, \tilde{k}_{ji}, \tilde{\sigma}'_{ji})$, where $\tilde{\vec{t}}_{ji} = (\tilde{t}^1_{ji}, \tilde{t}^2_{ji}, \dots, \tilde{t}^{\mathrm{R}}_{ji})$, $\tilde{\vec{r}}_{ji} = (\tilde{r}^1_{ji}, \tilde{r}^2_{ji}, \dots, \tilde{r}^{\mathrm{R}}_{ji})$ and $\tilde{\vec{\sigma}}_{ji} = (\tilde{\sigma}^1_{ji}, \tilde{\sigma}^2_{ji}, \dots, \tilde{\sigma}^{\mathrm{R}}_{ji})$, such that* $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathsf{vk}_j, u, P_i, \tilde{t}^u_{ji}, \tilde{r}^u_{ji}, \tilde{\sigma}^u_{ji}) = 0$ *for some u or* $\mathrm{Vrfy}_{\mathrm{SIG}}(\mathsf{vk}_j, P_i, \tilde{k}_{ji}, \tilde{\sigma}'_{ji}) = 0$*, or decrypts to a tuple $(P', \tilde{\vec{t}}_{ji}, \tilde{\vec{r}}_{ji}, \tilde{\vec{\sigma}}_{ji}, \tilde{k}_{ji}, \tilde{\sigma}'_{ji})$ where $P' \neq P_j$, or $\tilde{c}$ does not decrypt correctly.*

  If this does not hold, send $\perp$ to $\mathcal{F}^{\mathtt{reac}}_{\mathcal{G}}$. Else, send all verification keys $(\mathsf{vk}_1, \dots, \mathsf{vk}_N)$ as well as all tuples $(\hat{t}^1_{ji}, \hat{r}^1_{ji}, \hat{\sigma}^1_{ji}, \hat{k}_{ji}, \hat{\sigma}'_{ji})$ $(j \in \{1, \dots, N\}$ and $(\tilde{x}^1_i, \tau^1_i)$ to $\mathcal{F}^{\mathtt{reac}}_{\mathcal{G}}$.

- *Instruct the* IIM *to* connect *its air-gap switch to $\mathcal{Z}$.*

Subsequent Online Compute Phases
Upon receiving an input $x^u_i \in \{0,1\}^p$ in round u, each IIM *does the following:*

IIM: *Initially, set $u = 2$. Compute $\tilde{x}^u_i = x^u_i + t^u_i$ and $\tau^u_i \leftarrow \mathrm{Mac}(k_i, \mathtt{Inp\ Round\ u}, \tilde{x}^u_i)$ and send $(\tilde{x}^u_i, \tau^u_i)$ to one's main party. Increment u.*

- *Party P_i then sends $(\hat{t}^u_{ji}, \hat{r}^u_{ji}, \hat{\sigma}^u_{ji})$ $(j \in \{1, \dots, N\}$ and $(\tilde{x}^u_i, \tau^u_i)$ to $\mathcal{F}^{\mathtt{reac}}_{\mathcal{G}}$.*

Online Output Phases
Having completed its last step in the compute phase in round u, a party P_i requests output from $\mathcal{F}^{\mathtt{reac}}_{\mathcal{G}}$ for round u and forwards that output to OIM.

OIM: *Initially, set $u = 1$ and store the first input $(k_i, (r^1_i, \dots, r^R_i))$ from one's main party. On subsequent inputs (o^u_i, θ^u_i) or $\perp$ from one's main party, do the following: If the received value equals $\perp$, output $\perp$. Otherwise, check if* $\mathrm{Vrfy}_{\mathrm{MAC}}(k_i, o^u_i, \theta^u_i) = 1$ *and output $y^u_i = o^u_i + r^u_i$ if this holds, and $\perp$ otherwise. Always increment u.*

We are now ready to state our theorem for reactive functionalities. The proof is similar to the proof of Theorem 5.16 and therefore omitted.

Theorem 5.22 (Up to $N-1$ Corrupted Parties, Reactive Case)**.** *Let $\mathcal{G}$ be a reactive standard adaptively well-formed functionality. Let* PKE *and* SIG *be as in Theorem 5.16 and assume that* MAC *is EUF-CMA-secure. Then it holds that*

$$\Pi_{\mathcal{G}}^{(3)} \underset{\#\#}{\geq} \mathtt{AG}([\mathcal{G}])$$

for up to $N-1$ parties under adversarial control.

5.5.1 Up to N Parties Under Adversarial Control

With the same augmentation as described in Section 5.4.1, one can obtain a protocol $\Pi_{\mathcal{G}}^{(4)}$ that is also secure if the adversary corrupts *all* parties (cf. Fig. 5.7). We again omit the proof.

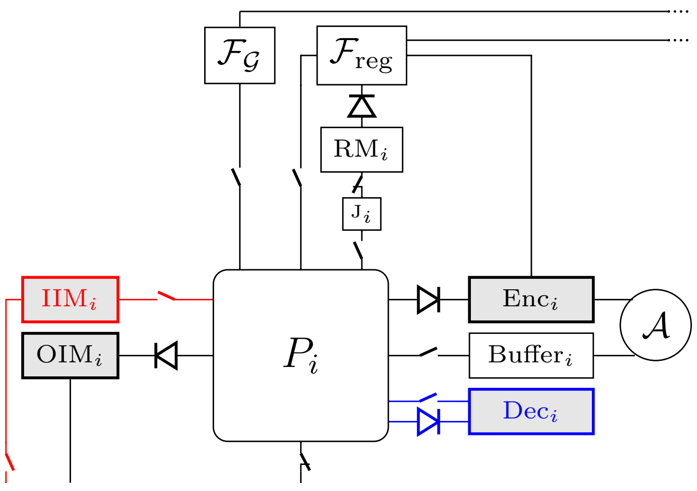


Figure 5.7: Architecture of $\Pi_{\mathcal{G}}^{(4)}$: Each party P_i ($i = 1, \dots, N$) has 3 hackable sub-parties, called *buffer*, *registration module* (RM) and *join* (J), and 4 unhackable sub-parties, called Enc(-unit), Dec(-unit), OIM and IIM. Buffer and Enc-unit are connected to the adversary via standard channels. All air-gap switches, except for P's air-gap switch to the environment and the RM's air-gap switch to J, are initially *disconnected.*

Theorem 5.23 (Up to N Corrupted Parties, Reactive Case)**.** *Let $\mathcal{G}$ be a reactive standard adaptively well-formed functionality. Let* PKE, SIG, MAC *be as in Theorem 5.22. Then it holds that*

$$\Pi_{\mathcal{G}}^{(4)} \underset{\#\#}{\geq} \mathtt{AG}([\mathcal{G}])$$

for up to N parties under adversarial control.

5.6 Architectures without Erasure

We can also obtain the results in Theorems 5.16 and 5.21 to 5.23 without relying on erasure by introducing an additional hackable interface module S_i that is connected to its main party P_i via a data diode and to the environment via an initially connected air-gap switch (cf. Fig. 5.8). S_i takes the (first) input, disconnects its air-gap switch to the environment, and carries out the sharing phase. Afterwards, S_i sends its own shares together with their signatures (and for reactive functionalities also MAC tags) and the verification key and secret key to P_i, who then carries out all further computations. S_i is then never activated again, remains offline throughout the protocol execution and thus cannot be corrupted though an `online-attack` instruction.[30] Note, however, that S_i can only be reused in subsequent protocols if it can be reset to its initial state. Such a reset is in line with what is implicitly assumed in large parts of the MPC literature, e.g. in the UC framework, where parties holding secrets cease to exist after protocol execution.

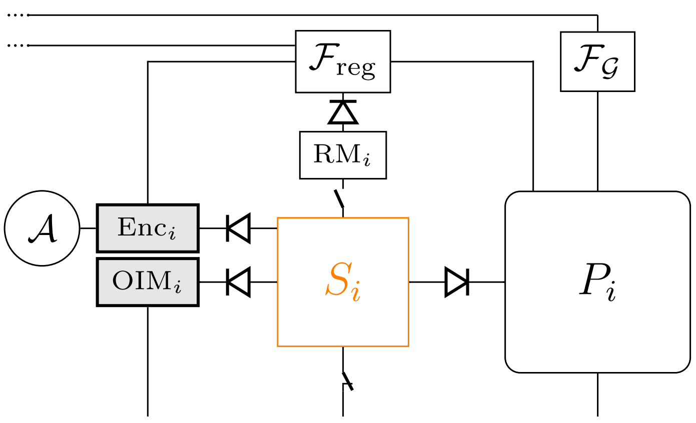


Figure 5.8: Architecture without Erasure (for up to $N-1$ Parties under Adversarial Control, Non-Reactive Case).

[30]Note that in order to formalize the security of this modified protocol, one has to add additional "dummy" interface modules (corresponding to the S_i's) in the ideal protocol $\texttt{AG}([\mathcal{G}])$ that are connected to $[\mathcal{G}]$ via a data diode and to the environment via an initially disconnected air-gap switch.

Chapter 6

Conclusion and Prospects

In this thesis, we provided two new security frameworks with a composable security notion along with several new general MPC protocols that were proven secure in one of these frameworks.

Our first framework is based on the idea of granting simulators only restricted access to the results computed by a stateful super-polynomial oracle. In this framework, we constructed two constant-round general MPC protocols in the plain model based on standard polynomial-time assumptions: a non-black-box protocol based on enhanced trapdoor permutations and a fully black-box construction based on verifiable perfectly binding homomorphic commitment schemes and IND-CPA-secure PKE schemes with oblivious public-key generation.

Our second framework allows to adequately capture the advantages (e.g. isolation properties) provided by remotely unhackable hardware modules. Utilizing only very few and very simple remotely unhackable hardware modules, we constructed several general MPC protocols in this framework for realizing the fortified functionality of almost every (non-reactive or reactive) ideal functionality.

In the following, we briefly discuss some research questions that arise from this work.

Practical Round Complexity and Unconditional Security Guarantees. In Chapter 4, we constructed general MPC protocols in the plain model based on standard polynomial-time assumptions which—being constant-round—are much more round-efficient than the general MPC protocols based on standard polynomial-time assumptions in the UC with super-polynomial helpers framework (recall that all general MPC protocols in the plain model in the Angel-based security framework are based on non-standard or super-polynomial-time assumptions). However, the round complexity of our constructions is still impractically high. It is therefore desirable to construct general MPC protocols (in the plain model and based on standard polynomial time assumptions) with a low, practical round complexity as opposed to a merely constant round complexity.

Another open problem related to our work is to construct protocols in the plain model where some security guarantees hold *unconditionally.* This is inherently impossible to achieve for tasks such as oblivious transfer, commitments or zero-knowledge in a framework like the one presented in Chapter 4. For instance, perfectly binding commitment schemes cannot be constructed in our framework since simulators need to be able to break the binding property of the real protocol to circumvent the impossibility results of [CF01], which is impossible with only super-polynomial resources if the binding property holds information-theoretically. It is therefore desirable to find a framework providing a composable security notion that allows to construct protocols for the above-mentioned tasks with unconditional security guarantees.

We have already begun to investigate both of the above-mentioned problems. One approach we believe has potential is based on the idea of granting simulators not only super-polynomial resources but also *functions of the honest parties' secret inputs if these functions "hide" these inputs.* For instance, a simulator corrupting the receiver may obtain a (hiding) commitment to the secret value sent to $\mathcal{F}_{\text{com}}$ by the honest sender. This way, the simulator can trivially simulate successfully even if the real protocol is perfectly binding. Also, there is no need to build an equivocation trapdoor into the commitment protocol which increases the overall round complexity as in the commitment protocol presented in Chapter 4.

Although the above approach is intuitively appealing, it comes with several problems that are yet to be addressed. First, giving the simulator values depending on secret inputs may result in trivial, meaningless security notions. For instance, a commitment scheme that is hiding but not binding could still be argued secure. This is because the simulator's ability to send the environment a commitment to the honest sender's input value essentially implies the tautology that the real protocol emulates the ideal protocol because the former is also executed in the latter. To address this problem, one must find a security notion based on the above approach that can be properly justified (note that we achieved such a justification for the framework in Chapter 4 by proving that its security notion implies the well-studied notion of SPS security). Another problem is that a security notion based on the above approach is inherently "non-monotonic". This means that one can construct protocols which are secure given an assumption A holds and another assumption B does *not* hold but which become completely insecure if *both* A and B hold. Identifying in which classes of cryptographic tasks this problem can arise remains an open problem.

More Efficient Protocols with Strong Security Guarantees against Online Attacks. In Chapter 5, we have proven strong feasibility results against remote hacking attacks. However, the general MPC protocols constructed in this chapter are impractical due to the high computational overhead of our modular approach of sharing inputs and then subsequently computing the desired function with these shares. It is therefore desirable to find a different approach that leads to more efficient protocols achieving the same level of security as the constructions presented in this work.

We believe this is possible using a more low-level, non-modular approach. More specifically, an appropriate protocol architecture based on a careful analysis of, e.g., the protocol in [IPS08] could provide the desired result.

Bibliography

[AIR01] Bill Aiello, Yuval Ishai, and Omer Reingold. “Priced Oblivious Transfer: How to Sell Digital Goods.” In: *Advances in Cryptology – EUROCRYPT 2001: 20th Annual International Conference on the Theory and Application of Cryptographic Techniques, Proceedings.* Springer, 2001, pp. 119–135.

[AMR14] Dirk Achenbach, Jörn Müller-Quade, and Jochen Rill. “Universally Composable Firewall Architectures Using Trusted Hardware.” In: *BalkanCryptSec 2014*. Vol. 9024. LNCS. Springer, 2014, pp. 57–74.

[Bar+04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. “Universally Composable Protocols with Relaxed Set-Up Assumptions.” In: *45th Annual IEEE Symposium on Foundations of Computer Science*. FOCS ’04. IEEE. 2004, pp. 186–195.

[Bar+14] Joshua Baron, Karim El Defrawy, Joshua Lampkins, and Rafail Ostrovsky. “How to withstand mobile virus attacks, revisited.” In: *ACM Symposium on Principles of Distributed Computing, PODC ’14, Paris, France, July 15-18, 2014*. 2014, pp. 293–302.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. “Short Group Signatures.” In: *Advances in Cryptology – CRYPTO 2004: 24th Annual International Cryptology Conference, Proceedings.* Springer, 2004, pp. 41–55.

[BCG93] Michael Ben-Or, Ran Canetti, and Oded Goldreich. “Asynchronous secure computation.” In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*. 1993, pp. 52–61.

[Bel+98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. “Relations Among Notions of Security for Public-Key Encryption Schemes.” In: *Advances in Cryptology – CRYPTO 1998. Proceedings.* Springer. 1998, pp. 26–45.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. “Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract).” In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*. 1988, pp. 1–10.

[Blu81] Manuel Blum. “Coin Flipping by Telephone.” In: *Advances in Cryptology – CRYPTO 1981: IEEE Workshop on Communications Security.* University of California, Santa Barbara, Deptartment of Elecrical and Computer Engineering, 1981, pp. 11–15.

[Bro+17] Brandon Broadnax, Nico Döttling, Gunnar Hartung, Jörn Müller-Quade, and Matthias Nagel. “Concurrently Composable Security with Shielded Super-Polynomial Simulators.” In: *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I.* 2017, pp. 351–381.

[Bro+18a] Brandon Broadnax, Valerie Fetzer, Jörn Müller-Quade, and Andy Rupp. “Non-malleability vs. CCA-Security: The Case of Commitments.” In: *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II.* 2018, pp. 312–337.

[Bro+18b] Brandon Broadnax, Matthias Huber, Bernhard Löwe, Jörn Müller-Quade, and Patrik Scheidecker. “Towards Efficient Software Protection Obeying Kerckhoffs’s Principle using Tamper-proof Hardware.” In: *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications, ICETE 2018 - Volume 2: SECRYPT, Porto, Portugal, July 26-28, 2018.* 2018, pp. 719–724.

[Bro+18c] Brandon Broadnax, Alexander Koch, Jeremias Mechler, Tobias Müller, Jörn Müller-Quade, and Matthias Nagel. “Fortified Universal Composability: Taking Advantage of Simple Secure Hardware Modules.” In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 519. URL: `https://eprint.iacr.org/2018/519`.

[BS05] Boaz Barak and Amit Sahai. “How to play almost any mental game over the net – concurrent composition via super-polynomial simulation.” In: *46st Annual IEEE Symposium on Foundations of Computer Science.* FOCS ’05. IEEE. 2005, pp. 543–552.

[BS99] Mihir Bellare and Amit Sahai. “Non-malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization.” In: *Advances in Cryptology – CRYPTO 1999. Proceedings.* Springer. 1999, pp. 519–536.

[Can+02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. “Universally Composable Two-party and Multi-party Secure Computation.” In: *Proceedings of the 34th Annual ACM Symposium on Theory of Computing.* STOC ’02. ACM, 2002, pp. 494–503.

[Can+07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. “Universally Composable Security with Global Setup.” In: *Theory of Cryptography: 4th Theory of Cryptography Conference, TCC 2007, Proceedings.* Springer, 2007, pp. 61–85.

[Can+96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. “Adaptively Secure Multi-Party Computation.” In: *STOC 1996.* 1996, pp. 639–648.

[Can00] Ran Canetti. “Security and composition of multiparty cryptographic protocols.” In: *Journal of CRYPTOLOGY* 13.1 (2000), pp. 143–202.

[Can01] Ran Canetti. “Universally Composable Security: A New Paradigm for Cryptographic Protocols.” In: *The 42th Annual IEEE Symposium on Foundations of Computer Science, 2001. Proceedings.* FOCS 2001. IEEE. 2001, pp. 136–145.

[Can06] Ran Canetti. *Security and Composition of Cryptographic Protocols: A Tutorial.* Cryptology ePrint Archive, Report 2006/465. `https://eprint.iacr.org/2006/465`. 2006.

[CDM00] Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. “General Secure Multi-party Computation from any Linear Secret-Sharing Scheme.” In: *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding.* 2000, pp. 316–334.

[CF01] Ran Canetti and Marc Fischlin. “Universally composable commitments.” In: *Advances in Cryptology – CRYPTO 2001: 21st Annual International Cryptology Conference, Proceedings.* Springer, 2001, pp. 19–40.

[CGJ15] Ran Canetti, Vipul Goyal, and Abhishek Jain. “Concurrent Secure Computation with Optimal Query Complexity.” In: *Advances in Cryptology – CRYPTO 2015: 35th Annual Cryptology Conference, Proceedings.* Springer, 2015, pp. 43–62.

[CGP15] Ran Canetti, Shafi Goldwasser, and Oxana Poburinnaya. “Adaptively Secure Two-Party Computation from Indistinguishability Obfuscation.” In: *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II.* 2015, pp. 557–585.

[CGT95] Claude Crépeau, Jeroen van de Graaf, and Alain Tapp. “Committed Oblivious Transfer and Private Multi-Party Computation.” In: *Advances in Cryptology - CRYPTO ’95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings.* 1995, pp. 110–123.

[Cha88] David Chaum. “The dining cryptographers problem: Unconditional sender and recipient untraceability.” In: *Journal of cryptology* 1.1 (1988), pp. 65–75.

[Cho+09] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. “Improved Non-committing Encryption with Applications to Adaptively Secure Protocols.” In: *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings.* 2009, pp. 287–302.

[Cho+18] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. “A Black-Box Construction of Non-malleable Encryption from Semantically Secure Encryption.” In: *J. Cryptology* 31.1 (2018), pp. 172–201.

[Cia+16] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. “Concurrent Non-Malleable Commitments (and More) in 3 Rounds.” In: *Advances in Cryptology – CRYPTO 2016. Proceedings.* Springer. 2016, pp. 270–299.

[Cia+17] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. “Four-Round Concurrent Non-Malleable Commitments from One-Way Functions.” In: *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II.* 2017, pp. 127–157.

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. “On the Limitations of Universally Composable Two-Party Computation without Set-up Assumptions.” In: *Advances in Cryptology – EUROCRYPT 2003: 22nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings.* Springer, 2003, pp. 68–86.

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. “Relaxing Chosen-Ciphertext Security.” In: *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings.* 2003, pp. 565–582.

[CLP10] Ran Canetti, Huijia Lin, and Rafael Pass. “Adaptive hardness and composable security in the plain model from standard assumptions.” In: *51st Annual IEEE Symposium on Foundations of Computer Science.* FOCS ’10. IEEE. 2010, pp. 541–550.

[CLP13] Ran Canetti, Huijia Lin, and Rafael Pass. “From Unprovability to Environmentally Friendly Protocols.” In: *54th Annual IEEE Symposium on Foundations of Computer Science.* FOCS ’13. IEEE. 2013, pp. 70–79.

[CPS07] Ran Canetti, Rafael Pass, and Abhi Shelat. “Cryptography from Sunspots: How to Use an Imperfect Reference String.” In: *48th Annual IEEE Symposium on Foundations of Computer Science.* FOCS ’07. IEEE. 2007, pp. 249–259.

[CPV17] Ran Canetti, Oxana Poburinnaya, and Muthuramakrishnan Venkitasubramaniam. “Equivocating Yao: constant-round adaptively secure multiparty computation in the plain model.” In: *STOC 2017.* ACM, 2017, pp. 497–509.

[CSV16] Ran Canetti, Daniel Shahaf, and Margarita Vald. “Universally Composable Authentication and Key-Exchange with Global PKI.” In: *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part II.* 2016, pp. 265–296.

[CVZ10] Zhenfu Cao, Ivan Visconti, and Zongyang Zhang. “Constant-Round Concurrent Non-Malleable Statistically Binding Commitments and Decommitments.” In: *Public Key Cryptography – PKC 2010.* Springer. 2010, pp. 193–208.

[Dac+09] Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Moti Yung. "Efficient Robust Private Set Intersection." In: *Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings.* 2009, pp. 125–142.

[Dac+13a] Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Muthuramakrishnan Venkitasubramaniam. "Adaptive and Concurrent Secure Computation from New Adaptive, Non-malleable Commitments." In: *Advances in Cryptology – ASIACRYPT 2013: 19th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings.* Springer, 2013, pp. 316–336.

[Dac+13b] Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Muthuramakrishnan Venkitasubramaniam. "Adaptive and Concurrent Secure Computation from New Adaptive, Non-malleable Commitments." In: *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I.* 2013, pp. 316–336.

[DDN00a] Danny Dolev, Cynthia Dwork, and Moni Naor. "Nonmalleable Cryptography." In: *SIAM Journal on Computing* 30.2 (2000), pp. 391–437.

[DDN00b] Danny Dolev, Cynthia Dwork, and Moni Naor. "Nonmalleable Cryptography." In: *SIAM J. Comput.* 30.2 (2000), pp. 391–437.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. "Non-Malleable Cryptography." In: *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing.* STOC 1991. ACM, 1991, pp. 542–552.

[DKR15] Dana Dachman-Soled, Jonathan Katz, and Vanishree Rao. "Adaptively Secure, Universally Composable, Multiparty Computation in Constant Rounds." In: *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II.* 2015, pp. 586–613.

[DN00] Ivan Damgård and Jesper Buus Nielsen. "Improved Non-committing Encryption Schemes Based on a General Complexity Assumption." In: *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings.* 2000, pp. 432–450.

[DN07] Ivan Damgård and Jesper Buus Nielsen. "Scalable and Unconditionally Secure Multiparty Computation." In: *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings.* 2007, pp. 572–590.

[Döt+13] Nico Döttling, Thilo Mie, Jörn Müller-Quade, and Tobias Nilges. "Implementing Resettable UC-Functionalities with Untrusted Tamper-Proof Hardware-Tokens." In: *TCC 2013.* Vol. 7785. LNCS. Springer, 2013, pp. 642–661.

[DS13] Ivan Damgård and Alessandra Scafuro. "Unconditionally secure and universally composable commitments from physical assumptions." In: *Advances in Cryptology – ASIACRYPT 2013: 19th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings*. Springer, 2013, pp. 100–119.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. "A randomized protocol for signing contracts." In: *Communications of the ACM* 28.6 (1985), pp. 637–647.

[Eld+18] Karim Eldefrawy, Rafail Ostrovsky, Sunoo Park, and Moti Yung. "Proactive Secure Multiparty Computation with a Dishonest Majority." In: *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*. 2018, pp. 200–215.

[ElG84] Taher ElGamal. "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms." In: *Advances in Cryptology – CRYPTO 1984: 4th Annual International Cryptology Conference, Proceedings*. Springer, 1984, pp. 10–18.

[Fei91] Uriel Feige. *Alternative Models for Zero Knowledge Interactive Proofs*. 1991.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. "Efficient Private Matching and Set Intersection." In: *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*. 2004, pp. 1–19.

[Gar+12] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. "Concurrently Secure Computation in Constant Rounds." In: *Advances in Cryptology – EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings*. Springer, 2012, pp. 99–116.

[Gar+15] Sanjam Garg, Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. "Cryptography with One-Way Communication." In: *CRYPTO 2015*. Vol. 9216. LNCS. Springer, 2015, pp. 191–208.

[Gar+16] Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. "The Exact Round Complexity of Secure Computation." In: *Advances in Cryptology – EUROCRYPT 2016. Proceedings Part II*. Springer. 2016, pp. 448–476.

[Ger+00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. "The Relationship between Public Key Encryption and Oblivious Transfer." In: *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*. 2000, pp. 325–335.

[GGJ13] Vipul Goyal, Divya Gupta, and Abhishek Jain. "What Information Is Leaked under Concurrent Composition?" In: *Advances in Cryptology – CRYPTO 2013: 33rd Annual Cryptology Conference, Proceedings*. Springer, 2013, pp. 220–238.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. “On the Cryptographic Applications of Random Functions.” In: *Advances in Cryptology, Proceedings of CRYPTO ’84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings.* 1984, pp. 276–288.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. “How to construct random functions.” In: *J. ACM* 33.4 (1986), pp. 792–807.

[GJ04] Philippe Golle and Ari Juels. “Dining Cryptographers Revisited.” In: *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings.* 2004, pp. 456–473.

[GJ13] Vipul Goyal and Abhishek Jain. “On Concurrently Secure Computation in the Multiple Ideal Query Model.” In: *Advances in Cryptology – EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings.* Springer, 2013, pp. 684–701.

[GJO10] Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. “Password-Authenticated Session-Key Generation on the Internet in the Plain Model.” In: *Advances in Cryptology – CRYPTO 2010: 30th Annual Cryptology Conference, Proceedings.* Springer, 2010, pp. 277–294.

[GK90] Oded Goldreich and Hugo 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.* 1990, pp. 268–282.

[GKP17] Sanjam Garg, Susumu Kiyoshima, and Omkant Pandey. “On the Exact Round Complexity of Self-composable Two-Party Computation.” In: *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II.* 2017, pp. 194–224.

[GKP18] Sanjam Garg, Susumu Kiyoshima, and Omkant Pandey. “A New Approach to Black-Box Concurrent Secure Computation.” In: *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II.* 2018, pp. 566–599.

[GM82] Shafi Goldwasser and Silvio Micali. “Probabilistic encryption & how to play mental poker keeping secret all partial information.” In: *Proceedings of the fourteenth annual ACM symposium on Theory of computing.* ACM. 1982, pp. 365–377.

[GM84] Shafi Goldwasser and Silvio Micali. “Probabilistic Encryption.” In: *J. Comput. Syst. Sci.* 28.2 (1984), pp. 270–299.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. “How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority.” In: *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*. 1987, pp. 218–229.

[GO96] Oded Goldreich and Rafail Ostrovsky. “Software protection and simulation on oblivious RAMs.” In: *Journal of the ACM (JACM)* 43.3 (1996), pp. 431–473.

[Gol03] Oded Goldreich. *Basic Tools*. Cambridge University Press, 2003.

[Gol04] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2004.

[Gol87] Oded Goldreich. “Towards a theory of software protection and simulation by oblivious RAMs.” In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. ACM. 1987, pp. 182–194.

[Goy+10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. “Founding Cryptography on Tamper-Proof Hardware Tokens.” In: *TCC 2010*. Vol. 5978. LNCS. Springer, 2010, pp. 308–326.

[Goy+12] Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. “Constructing Non-malleable Commitments: A Black-Box Approach.” In: *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*. 2012, pp. 51–60.

[Goy+14] Vipul Goyal, Silas Richelson, Alon Rosen, and Margarita Vald. “An Algebraic Approach to Non-malleability.” In: *55th Annual IEEE Symposium on Foundations of Computer Science*. FOCS ’14. IEEE. 2014, pp. 41–50.

[Goy+15] Vipul Goyal, Huijia Lin, Omkant Pandey, Rafael Pass, and Amit Sahai. “Round-Efficient Concurrently Composable Secure Computation via a Robust Extraction Lemma.” In: *Theory of Cryptography: 12th Theory of Cryptography Conference, TCC 2015, Proceedings*. Springer, 2015, pp. 260–289.

[Goy11] Vipul Goyal. “Constant Round Non-malleable Protocols Using One Way Functions.” In: *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*. STOC 2011. ACM. 2011, pp. 695–704.

[GP15] Sanjam Garg and Antigoni Polychroniadou. “Two-Round Adaptively Secure MPC from Indistinguishability Obfuscation.” In: *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*. 2015, pp. 614–637.

[GPR16] Vipul Goyal, Omkant Pandey, and Silas Richelson. “Textbook Non-malleable Commitments.” In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2016. ACM, 2016, pp. 1128–1141.

[HEK12] Yan Huang, David Evans, and Jonathan Katz. “Private Set Intersection: Are Garbled Circuits Better than Custom Protocols?” In: *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. 2012.

[HLP15] Carmit Hazay, Yehuda Lindell, and Arpita Patra. “Adaptively Secure Computation with Partial Erasures.” In: *PODC 2015*. ACM, 2015, pp. 291–300. URL: `http://dl.acm.org/citation.cfm?id=2767386`.

[HPV17] Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. “Constant Round Adaptively Secure Protocols in the Tamper-Proof Hardware Model.” In: *PKC 2017*. Vol. 10175. LNCS. Springer, 2017, pp. 428–460.

[HV15] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. “On Black-Box Complexity of Universally Composable Security in the CRS Model.” In: *Advances in Cryptology – ASIACRYPT 2015: 21st International Conference on the Theory and Application of Cryptology and Information Security, Proceedings, Part II*. Springer, 2015, pp. 183–209.

[HV16] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. “Composable Adaptive Secure Protocols without Setup under Polytime Assumptions.” In: *Theory of Cryptography: 14th Theory of Cryptography Conference, TCC 2016-B, Proceedings*. Printed version not yet published. 2016.

[IL89] Russell Impagliazzo and Michael Luby. “One-way Functions are Essential for Complexity Based Cryptography.” In: *The 30th Annual Symposium on Foundations of Computer Science, 1989. Proceedings*. FOCS 1989. IEEE. 1989, pp. 230–235.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. “Founding Cryptography on Oblivious Transfer – Efficiently.” In: *Advances in Cryptology – CRYPTO 2008: 28th Annual International Cryptology Conference, Proceedings*. Springer, 2008, pp. 572–591.

[IPS09] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. “Secure Arithmetic Computation with No Honest Majority.” In: *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*. 2009, pp. 294–314.

[Ish+06] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. “Black-box constructions for secure computation.” In: *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*. 2006, pp. 99–108.

[Kat07] Jonathan Katz. “Universally Composable Multi-party Computation Using Tamper-Proof Hardware.” In: *Advances in Cryptology – EUROCRYPT 2007: 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings*. Springer, 2007, pp. 115–128.

[Kiy14] Susumu Kiyoshima. “Round-Efficient Black-Box Construction of Composable Multi-Party Computation.” In: *Advances in Cryptology – CRYPTO 2014: 34th Annual Cryptology Conference, Proceedings.* Springer, 2014, pp. 351–368.

[KL11] Dafna Kidron and Yehuda Lindell. “Impossibility Results for Universal Composability in Public-Key Models and with Fixed Inputs.” In: *Journal of Cryptology* 24.3 (2011), pp. 517–544. Cryptology ePrint Archive (IACR): Report 2007/478. Version 2010-06-06.

[KLP07] Tauman Yael Kalai, Yehuda Lindell, and Manoj Prabhakaran. “Concurrent Composition of Secure Protocols in the Timing Model.” In: *Journal of Cryptology* 20.4 (Oct. 2007), pp. 431–492.

[KMO14] Susumu Kiyoshima, Yoshifumi Manabe, and Tatsuaki Okamoto. “Constant-Round Black-Box Construction of Composable Multi-Party Computation Protocol.” In: *Theory of Cryptography: 11th Theory of Cryptography Conference, TCC 2014, Proceedings.* Springer, 2014, pp. 343–367.

[Lin+15] Yehuda Lindell, Benny Pinkas, Nigel P. Smart, and Avishay Yanai. “Efficient Constant Round Multi-party Computation Combining BMR and SPDZ.” In: *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II.* 2015, pp. 319–338.

[Lin03] Yehuda Lindell. “General Composition and Universal Composability in Secure Multi-party Computation.” In: *44th Annual IEEE Symposium on Foundations of Computer Science.* FOCS ’03. IEEE. 2003, pp. 394–403.

[Lin04] Yehuda Lindell. “Lower Bounds for Concurrent Self Composition.” In: *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings.* 2004, pp. 203–222.

[Lin08] Andrew Y Lindell. “Efficient fully-simulatable oblivious transfer.” In: *Topics in Cryptology–CT-RSA 2008.* Springer, 2008, pp. 52–70.

[Lin09] Andrew Y Lindell. “Adaptively secure two-party computation with erasures.” In: *Cryptographers’ Track at the RSA Conference.* Springer. 2009, pp. 117–132.

[Lin13] Yehuda Lindell. “Fast Cut-and-Choose Based Protocols for Malicious and Covert Adversaries.” In: *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II.* 2013, pp. 1–17.

[Lin16] Huijia Lin. “Indistinguishability Obfuscation from Constant-Degree Graded Encoding Schemes.” In: *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I.* 2016, pp. 28–57.

[LK14] Yehuda Lindell and Jonathan Katz. *Introduction to modern cryptography.* Chapman and Hall/CRC, 2014.

[LOS14] Enrique Larraia, Emmanuela Orsini, and Nigel P. Smart. “Dishonest Majority Multi-Party Computation for Binary Circuits.” In: *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*. 2014, pp. 495–512.

[LP09] Huijia Lin and Rafael Pass. “Non-malleability Amplification.” In: *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*. STOC ’09. ACM, 2009, pp. 189–198.

[LP11] Huijia Lin and Rafael Pass. “Constant-round non-malleable commitments from any one-way function.” In: *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*. 2011, pp. 705–714.

[LP12] Huijia Lin and Rafael Pass. “Black-Box Constructions of Composable Protocols without Set-Up.” In: *Advances in Cryptology – CRYPTO 2012: 32nd Annual Cryptology Conference, Proceedings*. Springer, 2012, pp. 461–478.

[LPV08] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. “Concurrent Non-malleable Commitments from Any One-Way Function.” In: *Theory of Cryptography Conference*. TCC 2008. Springer, 2008, pp. 571–588.

[LPV09] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. “A Unified Framework for Concurrent Security: Universal Composability from Stand-alone Non-malleability.” In: *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*. STOC ’09. ACM, 2009, pp. 179–188.

[LPV12] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. “A Unified Framework for UC from Only OT.” In: *Advances in Cryptology – ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings*. Springer, 2012, pp. 699–717.

[LT17] Huijia Lin and Stefano Tessaro. “Indistinguishability Obfuscation from Trilinear Maps and Block-Wise Local PRGs.” In: *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*. 2017, pp. 630–660.

[MMY06] Tal Malkin, Ryan Moriarty, and Nikolai Yakovenko. “Generalized Environmental Security from Number Theoretic Assumptions.” In: *Theory of Cryptography: 3rd Theory of Cryptography Conference, TCC 2006, Proceedings*. Springer, 2006, pp. 343–359.

[MPR06] Silvio Micali, Rafael Pass, and Alon Rosen. “Input-Indistinguishable Computation.” In: *47th Annual IEEE Symposium on Foundations of Computer Science*. FOCS ’06. IEEE. 2006, pp. 367–378.

[MR91] Silvio Micali and Phillip Rogaway. “Secure Computation (Abstract).” In: *Advances in Cryptology - CRYPTO ’91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*. 1991, pp. 392–404.

[Nao89] Moni Naor. “Bit Commitment Using Pseudo-Randomness.” In: *Advances in Cryptology - CRYPTO ’89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings.* 1989, pp. 128–136.

[Nem17] Hamed Nemati. “Secure System Virtualization: End-to-End Verification of Memory Isolation.” PhD thesis. Royal Institute of Technology, Stockholm, Sweden, 2017. URL: `http://nbn-resolving.de/urn:nbn:se:kth:diva-213030`.

[NP01] Moni Naor and Benny Pinkas. “Efficient oblivious transfer protocols.” In: *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA.* 2001, pp. 448–457.

[OY91] Rafail Ostrovsky and Moti Yung. “How to Withstand Mobile Virus Attacks (Extended Abstract).” In: *PODC 1991.* ACM, 1991, pp. 51–59. URL: `http://dl.acm.org/citation.cfm?id=112600`.

[Pas03] Rafael Pass. “Simulation in Quasi-Polynomial Time, and Its Application to Protocol Composition.” In: *Advances in Cryptology – EUROCRYPT 2003: 22nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings.* Springer, 2003, pp. 160–176.

[PPV08] Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. “Adaptive One-Way Functions and Applications.” In: *Advances in Cryptology – CRYPTO 2008. Proceedings.* Springer, 2008, pp. 57–74.

[PR05] Rafael Pass and Alon Rosen. “Concurrent non-malleable commitments.” In: *46st Annual IEEE Symposium on Foundations of Computer Science.* FOCS ’05. IEEE. 2005, pp. 563–572.

[PR08] Manoj Prabhakaran and Mike Rosulek. “Cryptographic complexity of multi-party computation problems: Classifications and separations.” In: *Advances in Cryptology – CRYPTO 2008: 28th Annual International Cryptology Conference, Proceedings.* Springer, 2008, pp. 262–279.

[PR10] Benny Pinkas and Tzachy Reinman. “Oblivious RAM revisited.” In: *Advances in Cryptology–CRYPTO 2010.* Springer, 2010, pp. 502–519.

[PS04] Manoj Prabhakaran and Amit Sahai. “New Notions of Security: Achieving Universal Composability Without Trusted Setup.” In: *Proceedings of the 36th Annual ACM Symposium on Theory of Computing.* STOC 2004. ACM, 2004, pp. 242–251.

[PSV06] Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. “Construction of a Non-malleable Encryption Scheme from Any Semantically Secure One.” In: *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings.* 2006, pp. 271–289.

[PW09] Rafael Pass and Hoeteck Wee. “Black-Box Constructions of Two-Party Protocols from One-Way Functions.” In: *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings.* 2009, pp. 403–418.

[Qub18] Qubes OS Project. *Qubes Split GPG.* User Documentation. 2018. URL: `https://www.qubes-os.org/doc/split-gpg/` (visited on 05/08/2018).

[Rab81] Michael Rabin. *How To Exchange Secrets with Oblivious Transfer.* 1981.

[Rom90] John Rompel. “One-Way Functions are Necessary and Sufficient for Secure Signatures.” In: *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA.* 1990, pp. 387–394.

[RS91] Charles Rackoff and Daniel R. Simon. “Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack.” In: *Advances in Cryptology - CRYPTO ’91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings.* 1991, pp. 433–444.

[Shi+11] Elaine Shi, T-H Hubert Chan, Emil Stefanov, and Mingfei Li. “Oblivious RAM with O ((logN) 3) worst-case cost.” In: *Advances in Cryptology–ASIACRYPT 2011.* Springer, 2011, pp. 197–214.

[WC81] Mark N. Wegman and Larry Carter. “New Hash Functions and Their Use in Authentication and Set Equality.” In: *J. Comput. Syst. Sci.* 22.3 (1981), pp. 265–279.

[Wee10] Hoeteck Wee. “Black-Box, Round-Efficient Secure Computation via Non-malleability Amplification.” In: *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA.* 2010, pp. 531–540.

[WP+89] Michael Waidner, Birgit Pfitzmann, et al. “The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability.” In: *J.-J. Quisquater and J. Vandewalle, editors, Advances in Cryptology—EUROCRYPT* 89 (1989), p. 690.

[Yao82] Andrew Chi-Chih Yao. “Protocols for Secure Computations (Extended Abstract).” In: *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982.* 1982, pp. 160–164.

[Yao86] Andrew Chi-Chih Yao. “How to Generate and Exchange Secrets (Extended Abstract).” In: *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986.* 1986, pp. 162–167.

[ZGL18] Erkang Zheng, Phil Gates-Idem, and Matt Lavin. “Building a virtually air-gapped secure environment in AWS: with principles of devops security program and secure software delivery.” In: *Hot Topics in the Science of Security, HoTSoS 2018.* ACM, 2018, 11:1–11:8. URL: `http://dl.acm.org/citation.cfm?id=3190619`.

Own Publications

- Brandon Broadnax, Nico Döttling, Gunnar Hartung, Jörn Müller-Quade, and Matthias Nagel. “Concurrently Composable Security with Shielded Super-Polynomial Simulators.” In: *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I.* 2017, pp. 351–381
- Brandon Broadnax, Alexander Koch, Jeremias Mechler, Tobias Müller, Jörn Müller-Quade, and Matthias Nagel. “Fortified Universal Composability: Taking Advantage of Simple Secure Hardware Modules.” In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 519. URL: `https://eprint.iacr.org/2018/519`
- Brandon Broadnax, Valerie Fetzer, Jörn Müller-Quade, and Andy Rupp. “Non-malleability vs. CCA-Security: The Case of Commitments.” In: *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II.* 2018, pp. 312–337
- Brandon Broadnax, Matthias Huber, Bernhard Löwe, Jörn Müller-Quade, and Patrik Scheidecker. “Towards Efficient Software Protection Obeying Kerckhoffs's Principle using Tamper-proof Hardware.” In: *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications, ICETE 2018 - Volume 2: SECRYPT, Porto, Portugal, July 26-28, 2018.* 2018, pp. 719–724