Diss. ETH No. 18771

Exploring the Limits of Multi-Party Computation

A dissertation submitted to

ETH ZURICH

for the degree of
Doctor of Sciences

presented by

Dominik Raub
Dipl.-Inform., Universität Karlsruhe

born April 15, 1979, in Freudenstadt, Germany
citizen of Germany

accepted on the recommendation of

Prof. Dr. Ueli Maurer, examiner
Prof. Dr. Ronald Cramer, co-examiner

2009

Acknowledgments

First, I would like to thank my advisor Ueli Maurer, for his competent guidance and the valuable advice, both on a professional and personal level, that he generously provided throughout my doctoral studies.

Further, I would like to thank my co-referee Ronald Cramer, for taking the time to review my work.

Particular thanks also goes to my coauthors Jörn Müller-Quade, Robin Künzler, Christoph Lucas, and Matthias Fitzi without whom this thesis in its current form would probably not exist.

Also, I thank my colleagues, at ETH Zürich or elsewhere, Divesh Aggarwal, Zuzana Beerliova, Mark Fischlin, Peter Gazi, Martin Hirt, Thomas Holenstein, Renato Renner, Junji Shikata, Björn Tackmann, Alain Tapp, Stefano Tessaro, Stefan Wolf, Vassilis Zikas, and all those that I forgot to mention for many inspiring discussions.

Thanks also go to Beate Bernhard-Temme for her friendly and efficient handling of all administrative issues. For financial and logistical support I thank ETH Zürich and the Swiss National Science Foundation.

Last but not least I would like to thank my family, in particular my parents Mona and Wolfgang Raub, for their continuing support.

Abstract

In this thesis we explore the limits of multi-party computation and secure function evaluation. Secure function evaluation (SFE) protocols enable distrusting parties to compute an arbitrary function f on their joint inputs, without revealing anything besides the function output. In multi-party computation (MPC) protocols parties can additionally securely maintain a state and thus emulate an arbitrary stateful, reactive, possibly randomized functionality F.

SFE or MPC protocols need to tolerate a number of misbehaving participants that may try to learn secret information or to disrupt the computation. We generally make the worst case assumption that misbehaving participants are corrupted by a central adversary and collude. One distinguishes *computationally* (CO) secure protocols that rely on unproven computational assumptions and could in principle be broken by a very powerful adversary and *information theoretically* (IT) secure protocols which withstand even an unlimited attacker.

Unfortunately there are broad impossibility results for IT secure general MPC in presence of a majority of corrupted parties. These results motivate the following two lines of inquiry:

1. If *general* MPC guaranteeing full IT security against a corrupted majority is impossible, then what *specific* functions can still be computed in this setting?

2. If general MPC guaranteeing *full IT security* against a corrupted majority is impossible, is it at least possible to obtain a graceful degradation of security properties? Can we devise a protocol that is fully IT secure in case of few corruptions, but still provides CO security without fairness in case of many corruptions?

Pursuing the first line of inquiry we derive a combinatorial characterization of the functions computable with IT security in presence of arbitrarily many corrupted parties, in a number of adversarial and communication models.

Furthermore, we derive a combinatorial characterization of the functions computable with long-term security in a very natural, internet-like setting, where a network of insecure channels and a public-key infrastructure are available to the participants. For long-term security, we admit computational assumptions for the duration of a computation, but require IT security once the computation is concluded. Long-term security is a very interesting paradigm because the problem with CO assumptions is not so much that these could be unjustified right now, but rather that concrete CO assumptions could *eventually* be broken by an adversary whose power increases over time, e.g. due to new technical or algorithmic developments.

Following the second line of inquiry, we construct a hybrid secure MPC protocol that provides different levels of security depending on the number of corrupted parties. Our protocol allows for graceful degradation of security, from full IT security to weaker CO security, with an increasing number of corrupted parties, and is optimal under a number of bounds from the literature.

Finally, we discuss homomorphic encryption as a tool for non-interactive protocols. We show that there are no field-homomorphic one-way permutation (OWP), at least for fields of small characteristic. We conclude that field homomorphic cryptosystems are not obtainable using constructions along the lines of group homomorphic schemes, which are generally based on a group homomorphic OWP, e.g. RSA.

Zusammenfassung

In dieser Arbeit erkunden wir die Grenzen von Mehrparteienberechnungen (MPC) und sicherer Funktionsauswertung (SFE). Sichere Funktionsauswertungsprotokolle versetzen einander misstrauende Parteien in die Lage eine beliebige Funktion f ihrer Eingaben zu berechnen, ohne jenseits des Funktionsergebnisses etwas über ihre jeweiligen Eingaben preiszugeben. In Mehrparteienberechnungsprotokollen können Parteien zusätzlich auf sichere Weise einen Zustand halten und damit beliebige, zustandsbehaftete, möglicherweise randomisierte Funktionalitäten F emulieren.

SFE oder MPC Protokolle müssen tolerieren, dass eine gewisse Teilnehmer sich fehlverhalten, mit dem Ziel in den Besitz geheimer Informationen zu gelangen oder die Berechnung zu stören. Wir gehen gemeinhin vom schlechtesten anzunehmenden Fall aus, dass alle sich fehlverhaltenden Parteien von einem zentralen Gegner korrumpiert wurden und zusammenarbeiten. Man unterscheidet zwischen berechenmässig (CO) sicheren Protokollen, die auf unbewiesenen berechenmässigen Annahmen beruhen und prinzipiell von einem sehr mächtigen Gegner gebrochen werden könnten, und informationstheoretisch (IT) sicheren Protokollen, die sogar unbeschränkten Gegnern widerstehen.

Unglücklicherweise gibt es breite Unmöglichkeitsresultate für informationstheoretisch sichere allgemeine Mehrparteienberechnungen in Gegenwart einer Mehrheit von korrumpierten Parteien. Diese Resultate motivieren die folgenden zwei Fragestellungen:

1. Wenn *allgemeine* Mehrparteienberechnungen mit voller informationstheoretischer Sicherheit gegen eine korrumpierte Mehrheit unmöglich sind, welche *speziellen* Funktionen lassen sich dann in dieser Situation noch sicher berechnen?

2. Wenn allgemeine Mehrparteienberechnungen mit *voller informationstheoretischer Sicherheit* gegen eine korrumpierte Mehrheit unmöglich sind, ist es zumindest möglich eine graduelle Schächung der Sicherheitseigenschaften mit zunehmender Anzahl von korrumpierten Parteien zu erreichen? Können wir ein Protokoll angeben das im Fall weniger Korruptionen volle informationstheoretische Sicherheit bietet, aber immer noch berechenmässige Sicherheit gewährleistet, wenn viele Parteien korrumpiert sind?

Der ersten Fragestellung folgend leiten wir für verschiedene Gegner- und Kommunikationsmodelle eine kombinatorische Charakterisierung der Funktionen her, die mit informationstheoretischer Sicherheit in Gegenwart beliebig vieler korrumpierter Parteien berechenbar sind.

Weiterhin leiten wir eine kombinatorische Charakterisierung der Funktionen ab, die mit langfristiger Sicherheit berechenbar sind, in einem sehr natürlichen, internet-artigen Kommunikationsmodell, wo ein Netzwerk unsicherer Kanäle und eine Public-Key Infrastruktur gegeben sind. Langfristige Sicherheit bedeutet, dass wir berechenmässige Annahmen zulassen, aber nur während der Ausführung des Protokolls. Nach Abschluss der Berechnung fordern wir informationstheoretische Sicherheit. Langfristige Sicherheit ist ein sehr interessantes Paradigma, da das Problem mit berechenmässige Annahmen weniger darin besteht, dass sie zum jetzigen Zeitpunkt falsch sein könnten, als vielmehr darin, dass konkrete berechenmässige Annahmen schlussendlich von einem Gegner gebrochen werden könnten, der mit der Zeit mächtiger wird, e.g. auf Grund technischen oder algorithmischen Fortschritts.

Im Sinne der zweiten Fragestellung konstruieren wir hybrid-sichere Mehrparteienberechnungsprotokolle, welche verschiedene Sicherheitsgarantien bieten, abhängig davon wieviele Parteien korrumpiert sind. Unser Protokoll gewährleistet eine graduelle Schächung der Sicherheitseigenschaften mit steigender Anzahl korrumpierter Parteien, von voller informationstheoretischer Sicherheit zu schwacher, berechenmässige Sicherheit, und ist optimal unter Schranken aus der Literatur.

Zuletzt diskutieren wir homomorphe Verschlüsselung als ein Werkzeug für nicht-interaktive Protokolle. Wir zeigen, dass es keine körperhomomorphen Einwegpermutationen für Körper kleiner Charakteristik gibt. Wir folgern, dass körperhomomorphe Kryptosysteme nicht analog zu gruppenhomomorphen Systemen konstruierbar sind, welche gemeinhin auf gruppenhomomorphen Einwegpermutationen, e.g. RSA, basieren.

Contents

1 Introduction **11**

1.1 Multi-Party Computation and Secure Function Evaluation 11
1.2 On Secure Computability of Functions 13
1.3 On Optimally Hybrid-Secure MPC 14
1.4 On the Inexistence of Field-Homomorphic OWPs 15

2 Security Definitions and Notation **17**

2.1 Preliminaries 17
2.2 Computational Model 18
2.3 SFE and MPC 19
2.4 Communication Model and Corruption 20
2.5 Simulation-Based Security 20
2.6 UC Security 24
2.7 Stand-Alone Security 26
2.8 Ideal Functionalities 29

3 On Secure Computability of Functions **33**

3.1 Introduction 33
3.2 The Class $\mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$ of Passively Computable Functions 37
3.3 The Class $\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}}$ of Semi-Honestly Computable Functions . . 50
3.4 Symmetrization 61
3.5 The Class $\mathcal{F}_{\mathsf{act}}^{\mathsf{aut}}$ of Actively Computable Functions 63
3.6 Long-Term Security 75
3.7 Classification of 2-party Functions 81
3.8 Conclusions 83

4 On Optimally Hybrid-Secure MPC **87**
4.1 Introduction 87
4.2 Conventions and Simplifications 90
4.3 Hybrid-secure MPC: The Protocol π^{ρ} 91
4.4 Emulating a Party 95
4.5 Implementing a Designated Party MPC 100
4.6 Implementing a Hybrid-Secure MPC 109
4.7 The Security of the Stand-Alone Protocol π^{ρ}_{SA} 116
4.8 Protocols Without Broadcast Channel 117
4.9 Conclusions 118

5 On the Inexistence of Field-Homomorphic OWPs **119**
5.1 Introduction 119
5.2 The Representation Problem for Finite Black-Box Fields . . 124
5.3 The Representation Problem for $\mathbb{F}_{p^k}$ for Arbitrary Generators 130
5.4 Conclusions 136

6 Concluding Remarks **139**

Bibliography **143**

A Odds and Ends **153**
A.1 Perfectly Hiding or Perfectly Binding UC Commitments . 153

Chapter 1

Introduction

1.1 Multi-Party Computation and Secure Function Evaluation

In this thesis we explore the limitations of secure function evaluation (SFE) and multi-party computation (MPC). SFE was introduced by Yao [Yao82]: Given an arbitrary but fixed n-valued function f and a set of n mutually distrusting parties, each holding an input to the function f, an SFE protocol enables these parties to compute the function f on their joint inputs securely, even if some of the parties misbehave. Secure MPC is a generalization of SFE, enabling the secure computation of reactive and randomized functionalities that may maintain a state and repeatedly take input and give output.

We generally make the worst case assumption that misbehaving parties are corrupted by an adversary that coordinates their actions in order to discover the secrets of other parties or to disrupt the computation. We discuss *active* adversaries that may disrupt a protocol arbitrarily, as well as *semi-honest* and *passive* adversaries, that are limited to attempting to discover secret information while running the prescribed protocol. Passive adversaries are additionally constrained to operate with the provided protocol inputs, while semi-honest adversaries may substitute their protocol inputs with different ones in an effort to obtain extra information.[1]

[1] In the literature our notion of passive is occasionally referred to as semi-honest while our notion of semi-honest is sometimes referred to as *weakly* semi-honest or *weakly* passive.

Security requirements for MPC in the literature (e.g. [Gol01]) include privacy, correctness, robustness, fairness, and agreement on abort. *Privacy* is achieved if the adversary cannot learn more about the honest parties' inputs than what is implied by the inputs and outputs of the corrupted parties. *Correctness* means that the protocol output is correct for the given inputs according to the specification, or there is no output. In the following our notion of security generally encompasses these two basic requirements, privacy and correctness. Possible additional requirements are notions of output guarantees, which we discuss in order of decreasing strength: A protocol achieves *robustness* if an adversary cannot abort the computation and prevent the honest parties from obtaining output. *Fairness* is achieved if the honest parties obtain as much information about the output as the adversary. *Agreement on abort* means that all honest parties detect if one of them aborts (and then generally make no output). Security (privacy and correctness) with robustness is frequently referred to as *full security*. Security notions for MPC are discussed in more detail in Chapter 2.

MPC is generally discussed in a setting where parties are connected by a complete and synchronous network of secure or at least authenticated channels. Additionally an authenticated synchronous BC channel or a public key infrastructure (PKI) may be available. Universally composable (UC) MPC protocols usually also require a common reference string (CRS) [Can01, CF01].

A first general solution to the MPC problem was given by Goldreich et al. [GMW87], based on computational (CO) intractability assumptions and a broadcast (BC) channel. They achieve security with robustness against $t < \frac{n}{2}$ actively corrupted parties or with agreement on abort against $t < n$ actively corrupted parties as described in [Gol04]. Ben-Or et al. [BGW88], and independently Chaum et al. [CCD88], presented protocols which are information-theoretically (IT) secure and require no BC channel, but achieve full security only against $t < \frac{n}{3}$ actively corrupted parties. When no robustness is required (detectable MPC) [FHHW03] or if a BC channel is available [RB89], this bound can be improved to $t < \frac{n}{2}$.

Like the examples above, most MPC protocols in the literature are either IT secure, tolerating computationally unbounded adversaries, or CO secure, relying on unproven intractability assumptions and the limited computational power of the adversary. Invalidation of the underlying CO assumptions generally leads to a complete loss of security in CO secure protocols, even if only a single party is corrupted.

Naturally, IT full security is a desirable property for an MPC protocol. Unfortunately, IT fully secure general MPC tolerating an arbitrary number of actively or even passively corrupted parties is not achievable. This follows from a number of strong impossibility results for general MPC: Kilian [Kil00] shows that IT private general MPC cannot be obtained in presence of $t \geq \frac{n}{2}$ passively (or actively) corrupted parties. As shown by Cleve [Cle86], fairness cannot be achieved for general MPC in presence of $t \geq \frac{n}{2}$ actively corrupted parties. Ishai et al. [IKLP06] and Katz [Kat06] prove that there is a trade-off between robustness and privacy in MPC. They show that a protocol which guarantees robustness against up to $t \leq \rho$ actively corrupted parties can guarantee privacy against at most $t < n - \rho$ actively corrupted parties. The above impossibility results still hold even if an authenticated broadcast channel is provided.

These broad impossibility results motivate the following two lines of inquiry:

1. If *general* MPC guaranteeing full IT security against $t \geq \frac{n}{2}$ corrupted parties is impossible, then what *specific* functions can still be computed in this setting?
2. If general MPC guaranteeing *full IT security* against $t \geq \frac{n}{2}$ corrupted parties is impossible, is it at least possible to obtain a graceful degradation of security properties? Can we devise a protocol that is fully IT security in case of few corruptions, but still provides CO security without fairness in case of many corruptions?

We pursue the first line of inquiry in Chapter 3 on the secure computability of functions (as published in [KMQR09]). In Chapter 4 on hybrid-secure protocols we follow the second line of inquiry (as published in [LRM09]).

Finally we are also interested in non-interactive multi-party computation, where homomorphic encryption is an important tool. In Chapter 5 we show the inexistence of field-homomorphic one-way permutations (OWPs) and discuss implications for constructions of field-homomorphic encryption (as published in [MR07]).

1.2 On Secure Computability of Functions

In Chapter 3 we work towards characterizing which functions can be computed with full IT security in presence of $t \geq \frac{n}{2}$ corrupted parties.

The material presented in this chapter is published in [KMQR09].

We consider secure computability under active, semi-honest, and passive adversaries, possibly corrupting a majority of the protocol participants. From our findings we also derive a characterization of the functions computable with long-term (LT) security, where we admit computational assumptions for the duration of a computation, but require IT security (privacy) once the computation is concluded.

Long-term security is an interesting paradigm because a computationally unbounded adversary is not a realistic threat: The problem with CO assumptions is not so much that these could be unjustified right now, but rather that concrete CO assumptions could *eventually* be broken by an adversary whose power increases over time, e.g. due to new technical or algorithmic developments. LT secure protocols, in contrast to CO secure protocols, protect against such adversaries by relying on CO assumptions only during protocol execution.

We discuss computability of functions in the authenticated channels model with BC and in the public discussion model, where parties have access to a BC channel only. For LT secure computability we also analyze a very natural, internet-like setting, where a network of insecure channels and a public-key infrastructure are available to the participants.

1.3 On Optimally Hybrid-Secure MPC

In Chapter 4 we discuss hybrid-secure general MPC protocols. The material presented in this chapter is published in [LRM09]. Most MPC protocols are designed to be secure either against computationally unbounded IT adversaries or against weaker computationally bounded adversaries. IT MPC protocols tolerate unbounded adversaries but have the disadvantage that generally only a corrupted minority can be tolerated without compromising security. On the other hand, CO protocols can tolerate any number of corrupted parties if robustness and fairness are not required, but they are based on unproven intractability assumptions and the limited computational power of the adversary. Invalidation of the underlying assumptions generally leads to a complete loss of security, even if only a single party is corrupted. MPC protocols with hybrid security aim to combine the best of both worlds.

By providing different levels of security depending on the number of corrupted parties, hybrid secure MPC protocols allow for graceful degra-

dation of security, from IT full security to CO abort security, with increasing number of corrupted parties. We construct a UC-secure hybrid MPC protocol that combines IT and CO security and allows for an optimal trade-off between security with robustness, fairness, or agreement on abort, in a model where a complete secure synchronous network of secure channels with BC channel (and a CRS for the UC scenario) are available. For any fixed robustness parameter $\rho < \frac{n}{2}$ our protocol π_ρ simultaneously provides robust IT security in the presence of $t \leq \rho$ actively corrupted parties, fair IT security (no robustness) for $t < \frac{n}{2}$, and CO security with agreement on abort (no robustness) for $t < n - \rho$. As such our protocol is optimal under the bounds of Kilian [Kil00], Cleve [Cle86], Ishai et al. [IKLP06] and Katz [Kat06] as described above.

1.4 On the Inexistence of Field-Homomorphic One-Way Permutations

In Chapter 5 we show the inexistence of field-homomorphic one-way permutations (OWPs) for fields of small characteristic or in the non-uniform setting. The material presented in this chapter is published in [MR07]. We discuss implications for constructions of field-homomorphic encryption. Homomorphic encryption is an important tool in non-interactive MPC and server-aided secure computation.

We obtain our result on field-homomorphic OWPs by investigating black-box fields. Black-box fields serve to formalize representation independent, generic algorithms for finite fields. We discuss the problem of representing a given element of a black-box field, in terms of a prescribed set of generators for that field. We then find an efficient solution to this representation problem and deduce among other things the inexistence of field-homomorphic OWPs, at least for fields of small characteristic. We conclude that field-homomorphic cryptosystems are not obtainable using constructions along the lines of group-homomorphic schemes, e.g. RSA, that are generally constructed from group-homomorphic OWPs.

Note that even in the light of Gentry's [Gen09] recent construction for field-homomorphic public-key encryption our results are still relevant. Gentry presented a field-homomorphic public-key encryption scheme that allows evaluating circuits of depth linear in the size of the public key. Being based on a bootstrapping paradigm instead of field-homomorphic

OWPs, Gentry's result is not in contradiction to ours. However, providing a scheme where the size of the public key is independent of the circuit depth it can evaluate without resorting to non-standard assumptions is still an open problem. From our results one may at least conclude that such field-homomorphic cryptosystems are not obtainable using constructions along the lines of group-homomorphic schemes.

Chapter 2

Security Definitions and Notation

In the following we provide the formal groundwork for discussing *secure function evaluation* (SFE) and *multi-party computation* (MPC). We define the security of SFE and MPC protocols, both in the stand-alone (SA) and in the universally composable (UC) sense. We describe different adversarial and computational models, and formalize a number of security goals by giving appropriate ideal functionalities.

2.1 Preliminaries

Throughout this thesis, sets will generally be denoted by capitalized calligraphic letters, e.g. $\mathcal{X}$ or $\mathcal{Y}$. Notable exceptions are $\mathbb{C}$, $\mathbb{R}$, $\mathbb{Z}$, $\mathbb{N}$, and $\mathbb{P}$ denoting the complex, real, integer, natural, and positive prime numbers in $\mathbb{Z}$ respectively, and $\mathbb{F}_n$ denoting a finite field of size n.

Random variables will be denoted by capitals, e.g. $X : \Omega \to \mathcal{X}$, or $Y \sim_U \mathcal{Y}$ denotes a variable Y distributed uniformly over set $\mathcal{Y}$. Lower case letters will denotes values, e.g. $x \in \mathcal{X}$, $Y = y_1$. Given two random variables $X_A, X_B : \Omega \to \mathcal{X}$ with finite range $\mathcal{X}$, we denote by $\Delta(X_A, X_B)$ their statistical L1 distance

$$\Delta(X_A, X_B) = \frac{1}{2} \sum_{x \in \mathcal{X}} |\Pr_{X_A}(x) - \Pr_{X_B}(x)|.$$

Resources and functionalities will be denoted in sans-serif. For instance we will use BC^n for an n-party broadcast functionality, F_f for an ideal functionality computing a function f, A for an adversary, and S for a simulator.

2.2 Computational Model

In the following we will discuss and compose interactive algorithmic building blocks which we refer to as machines. We will consider different classes of machines, among them protocol machines, ideal functionalities, distiguishers, and simulators. Machines can be modelled in a variety of way, for instance as interactive turing machines, as in [Can01], or as state machines as in [BPW04]. All these models are generally equivalent, we largely follow [BPW04], with modifications motivated by the more abstract model of [Mau, Mau09]. We view machines as consisting of a number of named interfaces, an internal state, and algorithms acting on inputs at the interfaces, manipulating the internal state and possibly producing output to the interfaces. Each interface has a specified algorithm associated with it that is invoked when input is received at that interface and halts upon producing output (to the same or some other interface). Any two machines F_1 and F_2 can be composed into a new machine $\mathsf{F}_1 \circ \mathsf{F}_2$. Machine $\mathsf{F}_1 \circ \mathsf{F}_2$ is obtained by connecting the machines F_1 and F_2 by linking all interfaces that have the same name in both machines such that an output of machines F_1 at an interface a directly becomes an input to machines F_2 at interface a and vice versa. The interfaces of machine $\mathsf{F}_1 \circ \mathsf{F}_2$ are then the remaining interfaces of the machines F_1 and F_2. The running time of a machine is taken to be the sum of all steps ever made by its internal algorithms.

In order to arrive at a notion of computational security, we then need to fix when we consider an algorithm or a machine to be efficient or a quantity to be negligible.

Let $\kappa \in \mathbb{N}$ be a security parameter. Intuitively κ quantifies the level of security we expect to achieve in our protocols. Unless stated otherwise, asymptotic statements will be made in the security parameter κ.

We call a positive function $\epsilon : \mathbb{N} \to \mathbb{R}_{\geq 0}, \kappa \mapsto \epsilon(\kappa)$ *negligible* (in κ) if it approaches 0 faster than the inverse of any polynomial p:

$$\forall p \in \mathbb{R}[X]\ \exists \kappa_0 \in \mathbb{N}\ \forall \kappa \in \mathbb{N}\ :\ \kappa > \kappa_0 \implies \epsilon(\kappa) < \frac{1}{|p(\kappa)|}.$$

We call an algorithm or machine efficient, if its running time is polynomially bounded in the security parameter κ. We denote the class of efficient machines or algorithms by Poly and the class of all machines or algorithms by Algo.

2.3 SFE and MPC

In *secure function evaluation* (SFE) the goal is to compute a given function $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \to \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_n$ securely among a set $\mathcal{P} = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$ of n parties.[2] Each party $\mathsf{P}_i \in \mathcal{P}$ ($i \in [n] := \{1, \ldots, n\}$) holds an input $x_i \in \mathcal{X}_i$ from a set $\mathcal{X}_i$ and is supposed to receive output $f_i(x_1, \ldots, x_n) := y_i \in \mathcal{Y}_i$, where $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$. In *multi-party computation* (MPC) we are interested in implementing an arbitrary (possibly randomized, stateful, and reactive) n-party functionality F. The only restrictions on functionality F are that it provides an I/O-interface to each of the n parties and that it notifies the adversary of the length of any input or output, and the identity of the sending or receiving party P_i.[3] So in addition to performing SFE, the parties must be able to maintain a state across a computation encompassing several input and output (i.e. SFE) rounds. We extend the above notation to subsets $\mathcal{M} = \{P_{m_1}, \ldots, P_{m_{|\mathcal{M}|}}\} \subseteq \mathcal{P}$ of parties and denote their inputs by $x_{\mathcal{M}} := (x_{m_1}, \ldots, x_{m_{|\mathcal{M}|}})$ and their function outputs by $f_{\mathcal{M}}(x_1, \ldots, x_n) := y_{\mathcal{M}} := (y_{m_1}, \ldots, y_{m_{|\mathcal{M}|}})$. We call the set of all n-party functions $\mathcal{F}_n$ and the set of multi-party functions $\mathcal{F} := \bigcup_{n \geq 1} \mathcal{F}_n$.

In order to compute a function f or implement a functionality F parties $\mathsf{P}_i \in \mathcal{P}$ may execute a protocol π, utilizing resources (communication primitives) R. The resources R may for instance encompass a complete network of authenticated or secure channels, an authenticated broadcast channel or a functionality distributing a common reference string (CRS). A protocol $\pi = (\pi_1, \ldots, \pi_n)$ specifies a protocol machine π_i for every party $\mathsf{P}_i \in \mathcal{P}$, designed to communicate via the given resources R. As above we extend this notation to subsets $\mathcal{M} \subseteq \mathcal{P}$ of the parties and denote their protocol machines by $\pi_{\mathcal{M}}$. We generally demand that the protocol machines π_i to be run by parties $\mathsf{P}_i \in \mathcal{P}$ are efficient. We then designate by $\mathcal{H} \subseteq \mathcal{P}$ the set of honest parties that run their protocol machine π_i as

[2] In the 2-party setting we will occasionally refer to parties P_1 and P_2 as Alice or A and Bob or B.

[3] These restrictions apply, because we generally do not expect an implementation to hide who gives input or receives output, or the length of input or output.

specified by protocol π, and by $\mathcal{A} := \mathcal{P} \setminus \mathcal{H}$ the set of corrupted parties that may deviate from the protocol. We generally make the worst case assumption that corrupted parties are controlled by a central adversary A. The adversary (if present, i.e. if at least one party is corrupted) acts for the corrupted parties, sees messages sent over authenticated channels, and can manipulate messages sent over insecure channels. If no party is corrupted, we assume that no adversary is present. External adversaries that can listen on authenticated channels and manipulate insecure channels even when no party is corrupted are easily modelled by adding an additional party, that has no (or constant) output and whose inputs are ignored. *Security* then means that privacy of the inputs and correctness of the result should be guaranteed for the honest parties.

2.4 Communication Model and Corruption

In this thesis we consider SFE and MPC in a synchronous communication model with rushing adversaries and static corruption. Synchronous protocols proceed in well defined rounds. Messages sent in a particular round over reliable authenticated or secure channels arrive before the end of the round. Rushing adversaries may wait for all messages from honest parties before determining their own messages for the current round. Corruptions are static, meaning that we prove our protocol secure for any fixed set of corrupted parties, but not against adversaries that may adaptively corrupt parties during protocol execution.

As communication resources we will generally assume a complete network of reliable, synchronous authenticated or secure channels and a synchronous authenticated BC channel.

2.5 Simulation-Based Security

We use a simulation-based model of security, drawing heavily on the abstractions laid down by Maurer [Mau09, Mau]. The security of a protocol π running on resources R (the real setting) is assessed with respect to an ideal setting where a *trusted third party* (TTP) or ideal functionality F evaluates the prescribed function f or implements the desired interactive functionality. As such the TTP is used to specify all relevant aspects of the protocol π, in particular how inputs are handled, what is to be computed, what information may be leaked and how outputs are delivered.

A protocol π then achieves security according to the simulation paradigm if whatever an adversary A controlling a subset $\mathcal{A} \subseteq \mathcal{P}$ of parties can do in the real setting, a simulator (or ideal adversary) S could replicate in the ideal setting. This is formalized by means of an environment or distinguisher D. The distinguisher D provides inputs to and receives outputs from the protocol or TTP interfaces of the honest parties and the adversary A or the simulator S respectively. In the ideal setting the distinguisher D is run with the ideal functionality F, and the simulator S, where the simulator (or ideal adversary) S is connected to the interfaces of the corrupted parties to the ideal functionality F. We write $\mathsf{S}(\mathsf{A}) \circ \mathsf{F}$ for this ideal system. In the real setting the distinguisher D is run with protocol machines π_i ($i \in \mathcal{H}$) for the honest parties on communication resources R and adversary A connected to the interfaces of the corrupted parties to the communication resources R. We write $\mathsf{A} \circ \pi_{\mathcal{H}} \circ \mathsf{R}$ for this real system. The distinguisher D is expected to eventually output a decision bit $d \in \{0, 1\}$, that can be regarded as the distinguisher's guess whether it is connected to the real system $\mathsf{A} \circ \pi_{\mathcal{H}} \circ \mathsf{R}$ or the ideal system $\mathsf{S}(\mathsf{A}) \circ \mathsf{F}$. A protocol π achieves security in the simulation model if for any adversary A there is a simulator $\mathsf{S}(\mathsf{A})$ such that for any distinguisher D the real system $\mathsf{A} \circ \pi_{\mathcal{H}} \circ \mathsf{R}$ with adversary A and the ideal system $\mathsf{S}(\mathsf{A}) \circ \mathsf{F}$ with simulator $\mathsf{S}(A)$ are indistinguishable. Indistinguishability is formalized by means of the *advantage* that a distinguisher D has in distinguishing two system:

Definition 2.5.1 (Advantage of a Distinguisher)**.** The *advantage* of a distinguisher D in distinguishing systems S and S' is

$$\Delta^{\mathsf{D}}(S, S') := |\Pr[\mathsf{D}(S) = 0] - \Pr[\mathsf{D}(S') = 0]|,$$

where $\mathsf{D}(S)$ denotes the output of distinguisher D when interacting with system S. We can extend this notion to classes of distinguishers: The *advantage of a class* $\mathfrak{D}$ of distinguishers in distinguishing systems S and S' is

$$\Delta^{\mathfrak{D}}(S, S') := \sup_{\mathsf{D} \in \mathfrak{D}} |\Pr[\mathsf{D}(S) = 0] - \Pr[\mathsf{D}(S') = 0]|.$$

$\Diamond$

Simulation-based security can now be defined as follows:

Definition 2.5.2 (Simulation-Based Security)**.** Given a class of distinguishers $\mathfrak{D}$, a class of adversaries $\mathfrak{A}$, a class of simulators $\mathfrak{S}$, and an advantage function[4] $\epsilon(\kappa)$ (where κ is the security parameter) a protocol π

[4] usually a negligible function in the security parameter κ, or the 0-function

securely implements an ideal functionality F if for every adversary $\mathsf{A} \in \mathfrak{A}$ corrupting a set $\mathcal{A}$ of parties there is a simulator $\mathsf{S}(\mathsf{A}) \in \mathfrak{S}$ such that the advantage of distinguishing real model $\mathsf{A} \circ \pi_{\mathcal{H}} \circ \mathsf{R}$ and ideal model $\mathsf{S}(\mathsf{A}) \circ \mathsf{F}$ is bounded by the advantage function $\epsilon(\kappa)$:

$$\Delta^{\mathfrak{D}}(\mathsf{S}(\mathsf{A}) \circ \mathsf{F}, \mathsf{A} \circ \pi_{\mathcal{H}} \circ \mathsf{R}) \leq \epsilon(\kappa).$$

The resulting security paradigm is dependent on the choice of classes $\mathfrak{D}$, $\mathfrak{A}$, $\mathfrak{S}$, and the advantage function $\epsilon(\kappa)$, as well as the ideal functionality F. We will discuss several such security paradigms below. $\Diamond$

If a functionality F_2 can then be securely implemented given a functionality F_1 as sole resource, without employing any additional communication primitives or other resources, we say that functionality F_2 is securely locally reducible to functionality F_1:

Definition 2.5.3 (Local Reducibility)**.** If there exists a protocol π securely implementing a functionality F_2 given the set of resources $\mathsf{R} = \{\mathsf{F}_1\}$, as defined above, then we say functionality F_2 is *securely locally reducible* to functionality F_1 by means of protocol π.

If functionality F_1 is also securely locally reducible to functionality F_2, we say that functionalities F_1 and F_2 are *securely mutually locally reducible*. $\Diamond$

2.5.1 Universally Composable vs. Stand-Alone Security

We consider two basic types of simulation-based security, namely stand-alone (SA) security (see e.g. [Gol04]) and universally composable (UC) security [PW00, Can01, BPW04][5]. Both types of security can be derived from the basic definition of simulation-based security [Mau09].

SA security and UC security essentially differ in how the distinguisher D may interact with the protocol. A sufficient condition for composability [Mau09] are an absorbing class $\mathfrak{D}$ of distinguishers and the existence of a forwarding adversary in the class $\mathfrak{A}$ of adversaries: As the name suggests an forwarding adversary simply forwards all messages. Absorbing means that for a given protocol π, ideal functionality F and

[5] We follow the UC models of [Can01, BPW04] in spirit, but do not adhere to the notation of either work.

classes $\mathfrak{A}$ of adversaries, and $\mathfrak{S}$ of simulators, the composition of an adversary, simulator, protocol or functionality with a distinguisher D from the class $\mathfrak{D}$ must once again be a valid distinguisher $\mathsf{D}' \in \mathfrak{D}$:

$$\forall \mathsf{D} \in \mathfrak{D},\ \mathsf{A} \in \mathfrak{A},\ \mathsf{S} \in \mathfrak{S},\ \pi,\ \mathsf{F} :\quad \mathsf{D} \circ \mathsf{A},\ \mathsf{D} \circ \mathsf{S},\ \mathsf{D} \circ \pi,\ \mathsf{D} \circ \mathsf{F} \quad \in \mathfrak{D}.$$

Generally we demand that protocols, functionalities, and simulators are efficient, i.e. $\mathsf{S}, \pi, \mathsf{F} \in \mathsf{Poly}$, and obtain UC security by admitting all possible or all efficient distinguishers and adversaries, i.e. $\mathfrak{D} = \mathfrak{A} = \mathsf{Algo}$ or $\mathfrak{D} = \mathfrak{A} = \mathsf{Poly}$. The absorption property described above then allows for the derivation of a universal composition theorem [Mau09] along the lines of [PW00, Can01, BPW04].

If we restrict distinguishers to provide inputs before the protocol execution commences and receive outputs after the protocol terminates we lose universal composability and obtain the weaker notion of SA security.

Definition 2.5.4 (Stand-alone Distinguishers)**.** The class AlgoSA of stand-alone distinguishers consists of all distinguishing algorithms D that operate as follows: Distinguisher D provides input to the real or ideal system it is run with once, when it is first started. Furthermore, distinguisher D takes output from the real or ideal system it is run with once, before it makes its own output and terminates.

The subclass of efficient stand-alone distinguishers is denoted by PolySA. ◊

UC security and SA security admit different simplifications and as such we treat them in separate sections.

2.5.2 Computational and Information-Theoretic Security

Using the simulation based security paradigm above we can discuss both computational (CO) security and information-theoretic (IT) security [Mau09]. When only CO security is required it suffices to consider efficient distinguishers, adversaries and simulators ($\mathfrak{D}, \mathfrak{A}, \mathfrak{S} \subseteq \mathsf{Poly}$). For information-theoretic (IT) security on the other hand distinguishers, adversaries and simulators can be arbitrary unbounded algorithms ($\mathfrak{D}, \mathfrak{A}, \mathfrak{S} \subseteq \mathsf{Algo}$).

Furthermore, in the SA setting, we investigate long-term (LT) security where adversaries and simulators must be efficient but distinguishers may be unbounded ($\mathfrak{A}, \mathfrak{S} \subseteq \mathsf{Poly}$, $\mathfrak{D} \subseteq \mathsf{Algo}$), formalizing that we

expect CO assumptions to hold only for the duration of the protocol execution.

In all three cases the advantage function $\epsilon(\kappa)$ is chosen to be negligible in the security parameter κ. If in the IT case we fix an advantage of $\epsilon = 0$ we arrive at perfect (PF) security. We may vary the IT and PF cases by demanding efficient simulators (ITE, PFE).

Paradigm	$\mathfrak{D} =$	$\mathfrak{A} =$	$\mathfrak{S} =$	$\epsilon(\kappa)$	**Notation**
UC-PF	Algo	Algo	Algo	$\epsilon(\kappa) = 0$	$\pi \succcurlyeq^{\mathsf{UC-PF}} I$
UC-IT	Algo	Algo	Algo	$\epsilon(\kappa) < \mathsf{negl}$	$\pi \succcurlyeq^{\mathsf{UC-IT}} I$
UC-PFE	Algo	Algo	Poly	$\epsilon(\kappa) = 0$	$\pi \succcurlyeq^{\mathsf{UC-PFE}} I$
UC-ITE	Algo	Algo	Poly	$\epsilon(\kappa) < \mathsf{negl}$	$\pi \succcurlyeq^{\mathsf{UC-ITE}} I$
UC-CO	Poly	Poly	Poly	$\epsilon(\kappa) < \mathsf{negl}$	$\pi \succcurlyeq^{\mathsf{UC-CO}} I$
SA-PF	AlgoSA	Algo	Algo	$\epsilon(\kappa) = 0$	$\pi \succcurlyeq^{\mathsf{SA-PF}} I$
SA-IT	AlgoSA	Algo	Algo	$\epsilon(\kappa) < \mathsf{negl}$	$\pi \succcurlyeq^{\mathsf{SA-IT}} I$
SA-PFE	AlgoSA	Algo	Poly	$\epsilon(\kappa) = 0$	$\pi \succcurlyeq^{\mathsf{SA-PFE}} I$
SA-ITE	AlgoSA	Algo	Poly	$\epsilon(\kappa) < \mathsf{negl}$	$\pi \succcurlyeq^{\mathsf{SA-ITE}} I$
SA-CO	PolySA	Poly	Poly	$\epsilon(\kappa) < \mathsf{negl}$	$\pi \succcurlyeq^{\mathsf{SA-CO}} I$
SA-LT	AlgoSA	Poly	Poly	$\epsilon(\kappa) < \mathsf{negl}$	$\pi \succcurlyeq^{\mathsf{SA-LT}} I$

Table 2.1: Basic Security Paradigms

The resulting basic security paradigms are listed in Table 2.1. We may occasionally omit SA or UC from notation if the setting is clear from the context.

2.6 UC Security

We now take a closer look at UC security. Both the UC and SA definitions of security we use are simulation based. However, in contrast to SA definitions of security, in UC models of security, the distinguisher is not restricted to providing input and receiving output once. Rather the distinguisher may interact with the adversary or simulator during the entire course of the protocol execution. As such the distinguisher may run other protocols or copies of protocol π in parallel to the protocol execution of π in question and swap messages or otherwise use information from one protocol in another. The distinguisher can thus be thought of as the (possibly adversarial) environment in which the protocol π is running. This

allows for strong composition theorems [PW00, Can01, BPW04, Mau09]. A protocol π which UC securely implements an ideal functionality F can be used to securely replace the functionality F in *any* protocol context.

As shown in [Can01] under the term dummy adversary it is not necessary to consider a proper adversary A in the UC setting. Actually, the (real system) adversary A can be omitted [Mau09]. Its role is taken over by the distinguisher D, which then directly connects to the resource interfaces of the corrupted parties. Security is exhibited by providing a simulator S acting on the interfaces of the corrupted parties in the ideal setting that renders the ideal functionality with simulator $\mathsf{S}_{\mathcal{A}} \circ \mathsf{F}$ and the real protocol $\pi_{\mathcal{H}} \circ \mathsf{R}$ indistinguishable for any such distinguisher.

UC security then essentially states that wherever a protocol π is used, we can indistinguishably replace this protocol by the ideal functionality F it implements together with an appropriate simulator S and vice versa:

Definition 2.6.1 (Universally Composable (UC) Security)**.** Given a class $\mathfrak{D}$ of distinguishers, a class $\mathfrak{S}$ of simulators, and an advantage function $\epsilon(\kappa)$ (where κ is the security parameter) a protocol π UC-securely implements an ideal functionality F if for every set $\mathcal{A}$ of corrupted players there is a simulator $\mathsf{S}_{\mathcal{A}} \in \mathfrak{S}$ such that real system $\pi_{\mathcal{H}} \circ \mathsf{R}$ and ideal system $\mathsf{S}_{\mathcal{A}} \circ \mathsf{F}$ are indistinguishable:

$$\Delta^{\mathfrak{D}}(\mathsf{S}_{\mathcal{A}} \circ \mathsf{F}, \pi_{\mathcal{H}} \circ \mathsf{R}) \leq \epsilon(\kappa).$$

$\Diamond$

In the context of UC security, we generally take $\mathfrak{S}$ as the class of *efficient* simulators and the advantage function $\epsilon(\kappa)$ as a negligible function in the security parameter κ. If $\mathfrak{D}$ is the class of *efficient* distinguishers we obtain CO security, if $\mathfrak{D}$ is the class of *all* distinguishers we obtain ITE security.

Note that in contrast to [Can01] we use a synchronous communication model with static corruption, and we will generally assume a complete network of synchronous secure channels and a synchronous authenticated BC channel as resources. To avoid the impossibility results of [Can01, CF01], most of our work in the UC setting is in the CRS model, where a common reference string CRS drawn from a prescribed distribution is made available to all parties.[6] We will usually also present re-

[6] This setting can be viewed as a hybrid model, with ideal functionalities for reliable secure message transmission, BC, and CRS.

sults without a CRS for the SA setting. In the UC setting though, a correctly chosen CRS is a prerequisite to the security of protocols in the CRS model.[7]

2.7 Stand-Alone Security

The stand-alone (SA) model (see e.g. [Gol04]) differs from the UC model described above, in that distinguishers are taken from the class AlgoSA and thus restricted to interacting with the real or ideal system in the beginning and end of the protocol execution only, by providing inputs and receiving outputs respectively.

More precisely, in the SA model a distinguisher $\mathsf{D} \in \mathsf{AlgoSA}$ first provides inputs x_i ($\mathsf{P}_i \in \mathcal{H}$) for the honest parties and x_A for the adversary A. In the ideal setting the x_i ($\mathsf{P}_i \in \mathcal{H}$) are then input to the ideal functionality F, while input x_A is passed to the simulator S (which in turn computes inputs $x'_\mathcal{A}$ to F for the corrupted parties). In the real setting the protocol machines π_i are run on input x_i ($\mathsf{P}_i \in \mathcal{H}$), with the adversary A on input x_A and with resources R. Finally, in both settings the outputs $y_\mathcal{H}$ of the honest parties and y_A of the adversary or simulator are passed to the distinguisher D, which then has to output a decision bit $d \in \{0,1\}$, that can be regarded as the distinguisher's guess if it is connected to the real system $\mathsf{A} \circ \pi_\mathcal{H} \circ \mathsf{R}$ or the ideal system $\mathsf{S}(\mathsf{A}) \circ \mathsf{F}$. To facilitate a unified treatment of different corruption models, we will generally without loss of generality assume that $x_\mathsf{A} = (x_\mathcal{A}, x'_\mathsf{A})$ and $y_\mathsf{A} = (y_\mathcal{A}, y'_\mathsf{A})$ where $x_\mathcal{A} \in \mathcal{X}_\mathcal{A}$ and $y_\mathcal{A} \in \mathcal{Y}_\mathcal{A}$ are taken from the input and output spaces of the corrupted parties respectively, x'_A is an auxiliary input and y'_A is the protocol transcript observed by the adversary. This is indeed without loss of generality, as arbitrary inputs can be passed to the adversary via auxiliary input x'_A and whatever the adversary might compute from its observations can also be computed from the protocol transcript y'_A directly.

Once again a protocol π is now secure if for any adversary (from a specified class $\mathfrak{A}$) corrupting a certain set $\mathcal{A}$ of parties there is a simulator (from a specified class $\mathfrak{S}$) which renders ideal and real scenario indistinguishable for any distinguisher (from a specified class $\mathfrak{D}$).

[7] It is possible to minimize the reliance on the CRS such that our protocols tolerate an adversarially chosen CRS for few corrupted parties by applying techniques from [GK08, GO07] and a $(t, 2t-1)$-combiner for commitments (e.g. [Her05]). However, this construction is beyond the scope of this thesis.

Definition 2.7.1 (Stand-Alone Security)**.** Given a class $\mathfrak{D} \subseteq \mathsf{AlgoSA}$ of stand-alone distinguishers, a class $\mathfrak{A}$ of adversaries, a class $\mathfrak{S}$ of simulators, and an advantage function[8] $\epsilon(\kappa)$ (where κ is the security parameter) a protocol π SA-securely implements an ideal functionality F if for every adversary $\mathsf{A} \in \mathfrak{A}$ corrupting a set $\mathcal{A}$ of parties there is a simulator $\mathsf{S}(\mathsf{A}) \in \mathfrak{S}$ such that the advantage of distinguishing real model $\mathsf{A} \circ \pi_{\mathcal{H}} \circ \mathsf{R}$ and ideal model $\mathsf{S}(\mathsf{A}) \circ \mathsf{F}$ is bounded by the advantage function $\epsilon(\kappa)$:

$$\Delta^{\mathfrak{D}}(\mathsf{S}(\mathsf{A}) \circ \mathsf{F}, \mathsf{A} \circ \pi_{\mathcal{H}} \circ \mathsf{R}) \leq \epsilon(\kappa).$$

We can obtain different types of security, dependent on the choice of classes $\mathfrak{D}$, $\mathfrak{A}$, $\mathfrak{S}$, and advantage function $\epsilon(\kappa)$, and on the ideal functionality F. We will discuss several resulting security paradigms below. $\Diamond$

2.7.1 Adversarial Models

In the SA setting, we refine the paradigms CO, LT, and IT security further by defining adversarial models. An adversarial model is a restriction that can be imposed on the adversaries and simulators for any of the above paradigms. Restricted adversarial models are of interest because more restricted adversaries allow for a larger range of secure protocols, with possibly better properties (e.g. higher efficiency). Furthermore, a broader range of adversarial models allows for a more fine-grained insight into the nature of security.

A priori we consider *active* (act) adversaries that may misbehave in an arbitrary fashion, acting for the corrupted parties in any way they please. But a real world adversary may actually be weaker. For instance, we may have a *passive*[9] (pas) adversary that can learn the inputs, outputs and internal state of a corrupted party, e.g. by using bugs or sidechannels like electromagnetic radiation, without having the ability to interfere with protocol execution directly. Or we may consider a *semi-honest*[10] (sh) adversary that, like a passive adversary, does not interfere with protocol execution directly, but in addition to observing inputs, outputs and internal state of the corrupted parties, may also tamper with their inputs. Such behavior may occur because the adversary is interested to avoid

[8] usually a negligible function in the security parameter κ, or the 0-function

[9] In the literature our notion of passive is also occasionally referred to as semi-honest.

[10] In the literature our notion of semi-honest is also sometimes referred to as *weakly* semi-honest or *weakly* passive.

detection by adhering to the protocol, or because the actual protocol execution is secured by other, possibly physical, means. In particular we may apply computational tools like the protocol compiler of [GMW87] in an attempt to force active adversaries to behave passively. In such a scenario the adversary can unavoidably still substitute protocol inputs. The semi-honest setting models this scenario.

Naturally, we expect the same restrictions to apply to real and ideal adversaries (i.e. simulators). After all, a passively secure protocol should be as secure as the ideal functionality in presence of a passive adversary. However, note that in the case of semi-honest adversaries we can equivalently consider a more restricted real adversary, like in the passive case. Under the distinguisher classes $\mathfrak{D} \in \{\mathsf{PolySA}, \mathsf{AlgoSA}\}$ which we consider, we can for any distinguisher $\mathsf{D} \in \mathfrak{D}$ find a distinguisher $\mathsf{D}' = \mathsf{D} \circ \sigma \in \mathfrak{D}$ that incorporates the input substitution σ of a given adversary $\mathsf{A} = \mathsf{A}' \circ \sigma$. So we can find a passive adversary A' and a distinguisher D' that yield the same result as the original pair A and D.

In summary we arrive at the following

Definition 2.7.2 (Adversarial Models)**.** Active adversaries and the corresponding simulators may behave arbitrarily. We denote the class of active adversaries by $\mathfrak{A}_{\mathsf{act}}$, and the corresponding class of simulators by $\mathfrak{S}_{\mathsf{act}}$.

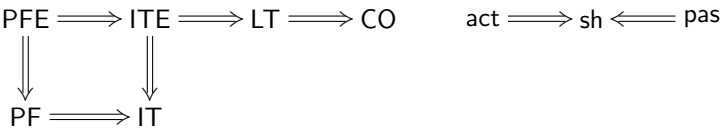
Semi-honest adversaries, collected in the class $\mathfrak{A}_{\mathsf{sh}}$, are restricted run the prescribed protocol π with the inputs $x_{\mathcal{A}}$ provided by the distinguisher D; the corresponding simulators, collected in the class $\mathfrak{S}_{\mathsf{sh}}$, may behave arbitrarily.

Passive adversaries, collected in the class $\mathfrak{A}_{\mathsf{pas}} = \mathfrak{A}_{\mathsf{sh}}$, are restricted in the same way as semi-honest adversaries and the corresponding simulators, collected in the class $\mathfrak{S}_{\mathsf{pas}}$, are restricted to forward the inputs $x_{\mathcal{A}}$ provided by the distinguisher D to the ideal functionality F. $\Diamond$

Security paradigms and adversarial models as defined above are combined by intersecting their defining sets, i.e. semi-honest SA-IT security is described by

$$\begin{aligned}
\mathfrak{D}_{\mathsf{sh}}^{\mathsf{SA-IT}} &= \mathfrak{D}^{\mathsf{SA-IT}} = \mathsf{AlgoSA}, \\
\mathfrak{S}_{\mathsf{sh}}^{\mathsf{SA-IT}} &= \mathfrak{S}^{\mathsf{SA-IT}} \cap \mathfrak{S}_{\mathsf{sh}} = \mathsf{Algo} \cap \mathfrak{S}_{\mathsf{sh}}, \\
\mathfrak{A}_{\mathsf{sh}}^{\mathsf{SA-IT}} &= \mathfrak{A}^{\mathsf{SA-IT}} \cap \mathfrak{A}_{\mathsf{sh}} = \mathsf{Algo} \cap \mathfrak{A}_{\mathsf{sh}}, \\
\epsilon(\kappa) &< \mathsf{negl}
\end{aligned}$$

and denoted $\pi \succcurlyeq_{\mathsf{sh}}^{\mathsf{SA}-\mathsf{IT}} I$. By definition we have the following implications among security paradigms and adversarial models respectively:



2.8 Ideal Functionalities

We will, in the following, generally be interested in securely implementing an n-party function f (SFE) or an n-party functionality F (MPC). The only restrictions on functionality F are that it provides an I/O-interface to each of the n parties and that it notifies the adversary of the length of any input or output, and the identity of the sending or receiving party P_i.

2.8.1 The SFE Functionality F_f

Securely implementing an n-party function f is formalized by means of a functionality F_f. Functionality F_f takes inputs $x_i \in \mathcal{X}_i$ from the parties $\mathsf{P}_i \in \mathcal{P}$. If a party P_i provides no input functionality F_f sets $x_i := x_i^{\mathsf{def}}$ for some fixed default input $x_i^{\mathsf{def}} \in \mathcal{X}_i$. Functionality F_f then computes $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ and outputs the y_i to the parties $\mathsf{P}_i \in \mathcal{P}$.

For finite functions f, where the input sets $\mathcal{X}_i$ and output sets $\mathcal{Y}_i$ are all finite, the functionality F_f as described above is already an n-party functionality. Explicit input and output notifications to the adversary are not required, because we can define functionality F_f such that each party gives input (possibly the default input by inactivity) of fixed length and receives output of fixed length in a fixed round. For infinite functions f, we have to add appropriate notifications to the definition of functionality F_f.

Securely implementing the function f then amounts to securely implementing the n-party functionality F_f. As such all statements for n-party functionalities F (e.g. those below) are easily applies to n-party functions f by setting $\mathsf{F} := \mathsf{F}_f$.

2.8.2 Implementing a Functionality with Weaker Guarantees

We now model implementing a functionality F with subsets of the security properties privacy, correctness, robustness, fairness, and agreement on abort introduced in Chapter 1. We describe ideal functionalities for the following specific security notions:

Full Security. Computing functionality F with *privacy, correctness and robustness*, which implies all the security notions mentioned above, is modelled by functionality F itself, since, in the setting which we consider, demanding a secure implementation of functionality F already amounts to demanding full security.

Fair Security. Demanding *privacy, correctness and fairness* (which implies agreement on abort) only for functionality F is captured by the ideal functionality $\mathsf{F}^{\mathsf{fair}}$, which operates as follows: $\mathsf{F}^{\mathsf{fair}}$ internally runs F. Any inputs to F are forwarded, as are any messages F may output to the adversary. If functionality F makes output $y_{\mathcal{M}} = (y_{m_1}, \ldots, y_{m_{|\mathcal{M}|}})$ to a subset $\mathcal{M} \subseteq \mathcal{P}$ of parties in a given round, then $\mathsf{F}^{\mathsf{fair}}$ request an output flag $o \in \{0, 1\}$ from the adversary, defaulting to $o = 1$ if the adversary makes no suitable input. Finally, for $o = 1$ functionality $\mathsf{F}^{\mathsf{fair}}$ makes output y_i to the parties $\mathsf{P}_i \in \mathcal{M}$, for $o = 0$ it makes output $\perp$ to *all* parties $\mathsf{P}_i \in \mathcal{P}$ and halts.

Abort Security. The functionality F^{ab}, specifying *privacy, correctness and agreement on abort* only, works like $\mathsf{F}^{\mathsf{fair}}$ but forwards the outputs $y_{\mathcal{M}\cap\mathcal{A}}$ of corrupted parties to the adversary before requesting an output flag.[11]

Designated Abort Security. Computing function f with a *designated aborter* (DA) is a notion weaker than fair security but stronger than abort security in that fairness is only guaranteed if a *designated party* P_d is honest. The functionality $\mathsf{F}^{\mathsf{des}}$, specifying *privacy, correctness and designated abort*, behaves like $\mathsf{F}^{\mathsf{fair}}$ if the designated party P_d is honest and like F^{ab} if the designated party P_d is corrupted.

No Security. The functionality $\mathsf{F}^{\mathsf{noSec}}$ models demanding no security whatsoever: Functionality $\mathsf{F}^{\mathsf{noSec}}$ turns control over to the adversary by forwarding all inputs from the honest parties to the adversary and letting the adversary fix all outputs to honest parties.

[11] We could relax the definition further by allowing the adversary to send one output flag for each party, dropping agreement on abort. However, all our protocols will achieve agreement on abort.

As a simulator $\mathsf{S}^{\mathsf{noSec}}$ can use the inputs of honest parties to simulate honest protocol machines, this already proves the following (rather trivial) lemma:

Lemma 2.8.1. *Any protocol π UC securely implements the ideal model $\mathsf{F}^{\mathsf{noSec}}$.*

2.8.2.1 On Relations between Security Properties

Note that for SFE in the 2-party setting (but not for $n > 2$ parties) robustness and fairness amount to the same. Given the fair functionality $\mathsf{F}_f^{\mathsf{fair}}$ we can implement the robust functionality F_f by having party P_i output $f_i(x_i, x_{2-i}^{\mathsf{def}})$ when it receives the abort signal $\perp$. Conversely, given the robust functionality F_f, we directly use it as implementation of fair functionality $\mathsf{F}_f^{\mathsf{fair}}$. The simulators are straightforward.

Lemma 2.8.2. *In the 2-party setting, the functionalities $\mathsf{F}_f^{\mathsf{fair}}$ and F_f are efficiently and PFE securely mutually locally reducible, even in presence of active adversaries, i.e. robustness and fairness amount to the same.*

Finally we show that computability by public discussion (authenticated BC only as resources R) and in the authenticated channels model (complete network of authenticated channels as resources R) lead to identical results for semi-honest or passive adversaries. In the authenticated channels model we can securely (against sh and pas adversaries) implement BC by simply sending messages to all other parties. Conversely in the authenticated BC model, authenticated channels can be implemented by broadcasting messages and instructing parties other than the intended recipient not to listen. By the last argument computability by public discussion and in the authenticated channels model *with BC* lead to identical results for active adversaries also.

Lemma 2.8.3. *In presence of semi-honest or passive adversaries, a functionality F is securely implementable in the authenticated channels model if and only if it is implementable by public discussion (authenticated BC only). In presence of active adversaries, a functionality F is securely implementable in the authenticated channels model* with BC *if and only if it is implementable by public discussion.*

Chapter 3

On Secure Computability of Functions

3.1 Introduction

As shown by Kilian [Kil00], IT secure general MPC cannot be attained in presence of $t \geq \frac{n}{2}$ passively (or actively) corrupted parties. In this chapter we work towards characterizing which *specific* functions can still be computed with full IT security in presence of $t \geq \frac{n}{2}$ corrupted parties. The material presented in this chapter is published in [KMQR09].

We consider secure computability under active, semi-honest, and passive adversaries, possibly corrupting a majority of the protocol participants. From our findings we also derive a characterization of the functions computable with long-term (LT) security, where we admit computational assumptions for the duration of a computation, but require IT security (privacy) once the computation is concluded.

Long-term security is an interesting paradigm because a computationally unbounded adversary is not a realistic threat: The problem with CO assumptions is not so much that these could be unjustified right now, but rather that concrete CO assumptions could *eventually* be broken by an adversary whose power increases over time, e.g. due to new technical or algorithmic developments. LT secure protocols in contrast to CO secure protocols, protect against such adversaries by relying on CO assumptions only during protocol execution.

All results in this chapter are constructive: whenever it is claimed that a class of functions is securely computable a protocol is given.

3.1.1 Contributions

We consider a synchronous communication model, where parties are provided with a complete network of authenticated channels and an authenticated BC channel. We then combinatorially characterize the classes of n-party functions securely computable by constant round protocols under CO unbounded passive, semi-honest, and active adversaries corrupting arbitrarily many parties.

We arrive at three classes of functions:

1. The class $\mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$ of *passively computable functions*, which can securely be computed by a constant round protocol in presence of a CO unbounded *passive*[12] adversary who must behave according to the protocol.

2. The class $\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}}$ of *semi-honestly computable functions*, and which can securely be computed by a constant round protocol in presence of a CO unbounded *semi-honest*[13] adversary that has to adhere to the protocol, but may replace his inputs.

3. The class $\mathcal{F}_{\mathsf{act}}^{\mathsf{aut}}$ of *actively computable functions*. which can securely be computed by a constant round protocol in presence of a CO unbounded *active* adversary which may disrupt the protocol arbitrarily.

We give a necessary and sufficient combinatorial condition for membership of a function f in the classes of *passively computable functions* $\mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$ and of *semi-honestly computable functions* $\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}}$. For membership of a function f in the class $\mathcal{F}_{\mathsf{act}}^{\mathsf{aut}}$ of *actively computable functions*, we give a sufficient condition, and prove necessity for two party functions. We only conjecture necessity for the multi-party case.

We then derive a characterization of the class $\mathcal{F}_{\mathsf{lts}}^{\mathsf{aut}}$ of *long-term securely computable functions*. Class $\mathcal{F}_{\mathsf{lts}}^{\mathsf{aut}}$ contains the functions f which can securely be computed by a constant round protocol in presence of an *active*

[12] In the literature our notion of passive is also occasionally referred to as semi-honest.

[13] In the literature our notion of semi-honest is also sometimes referred to as *weakly* semi-honest or *weakly* passive.

adversary corrupting arbitrarily many parties that remains CO bounded throughout the protocol execution, but may become unbounded thereafter. We show that the class $\mathcal{F}_{\mathsf{lts}}^{\mathsf{aut}}$ of LT securely computable functions equals the class of semi-honestly computable functions $\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} = \mathcal{F}_{\mathsf{lts}}^{\mathsf{aut}}$. Furthermore, we prove that the class of LT securely computable functions remains unchanged if we replace the authenticated channels model by the public discussion model or by a more realistic communication infrastructure of a network of insecure channels with a given public-key infrastructure (PKI). Hence our classification applies to a very practical, internet-like setting.

We show that the class $\mathcal{F}_{\mathsf{act}}^{\mathsf{aut}}$ of actively computable functions, constitutes a proper subset of the class $\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} = \mathcal{F}_{\mathsf{lts}}^{\mathsf{aut}}$ of semi-honestly or LT securely computable functions. So weakening security requirements from IT security to LT security allows for computing strictly more functions securely.

Unlike our IT secure protocols the LT secure protocols given in this work do not achieve robustness or fairness. We show that this is optimal in the sense that generally functions implementable with LT security cannot be implemented with fairness. However, we present protocols which guarantee that only a specific designated party can abort the computation after learning the output. I.e. the fairness property can only be violated by this designated party. Interestingly these protocols make use of CO secure oblivious transfer (OT) protocols even though OT itself cannot be achieved with LT security.

Summarizing our results and importing the treatment of complete two party functions from [KMQ08] (i.e. functions which are cryptographically as powerful as oblivious transfer) we arrive at a complete classification of two party functions. Interestingly, there is a class of functions which cannot securely be computed but still are not complete. This shows that for non-boolean functions there is no zero-one law for privacy like there is for boolean functions [CK89].

3.1.2 Related Work

Secure computability of functions was first discussed by [CK89]. They characterize the symmetric boolean functions (all parties receive the same output $y \in \{0,1\}$) that can be computed with IT security in presence of passive adversaries in the private channels model. In this scenario functions are either computable or complete (zero-one law for privacy).

Kushilevitz [Kus92] presented the first result for non-boolean functions describing the symmetric 2-party functions which can be computed with perfect security in presence of an unbounded passive adversary. Our protocols and proof techniques draw heavily upon [Kus92]. Also in the 2-party setting, [MQ05] sketches a generalization of [Kus92] to the asymmetric, IT case, connections to LT security and discusses quantum aspects, though without proper formalization or proofs. Our work goes beyond the results of [CK89, Kus92, MQ05] in that we consider IT secure computability of asymmetric, non-boolean functions, in presence of passive, semi-honest, active, and quantum adversaries, for the most part in the multi-party setting.

Gordon et al. [GHKL08] characterize the boolean functions computable with CO fairness in the 2-party setting in presence of active adversaries. Our protocols for active adversaries are robust (and hence fair) and being applicable to asymmetric, non-boolean functions, pertain to a larger class of functions than those of [GHKL08], but in the IT scenario instead of the CO setting.

Concurrently to our work Maji et al. [MPR09] developed a characterization of the symmetric 2-party functions computable under active or passive adversaries. Maji et al. do not discuss asymmetric or n-party functions ($n > 2$), but their characterization matches the restriction of ours to symmetric 2-party functions. In this restricted setting Maji et al. provide proofs that hold for computability under arbitrary (not only constant round) protocols. Furthermore, Maji et al. discuss questions of reducibility among symmetric 2-party functions.

Other works that deal with the computability of 2-party functions in the perfect or IT setting are [Kil91, Kil00, BMM99, KMQ08]. However, these papers focus mostly on reducibility and completeness, while we are more interested in computability in the authenticated channels model and implications for LT security. Computability of a few interesting special functions in presence of dishonest majorities is discussed in [BT07].

Everlasting security from temporary assumptions has been investigated in cryptographic research for some time. It was shown that a bound on the memory available to the adversary allows key exchange and OT protocols [CM97, CCM02] which remain secure even if the memory bound holds only during the execution of the protocol. This idea has been pursued further to achieve everlasting security from a network of distributed servers providing randomness [Rab03]. In [DM04] it was shown that using a CO secure key exchange in the bounded storage model need not yield everlasting security. For some time general

quantum cryptographic protocols were sought which obtain everlasting security from a temporary assumption. Such protocols are now generally accepted to be impossible [BCMS99]. Additional assumptions, like a temporary bound on the quantum memory can again provide everlasting security for secure computations [DFSS05].

In this paper we investigate the power of temporary CO assumptions in the standard model. This is along the lines of [MQU07]. However, in [MQU07] strong composability requirements are imposed under which little is possible without additional setup assumptions, like the temporary availability of secure hardware.

3.2 The Class $\mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$ of Passively Computable Functions

We subsequently characterize the class $\mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$ of n-party functions $f \in \mathcal{F}_n$ that are computable IT securely in the authenticated channels model in presence of a passive adversary.

Definition 3.2.1 ($\mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$: Passively Computable Functions)**.** The class of *passively computable* functions $\mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$ consists of the functions $f \in \mathcal{F}$ for which a constant round protocol $\pi \in \mathsf{Poly}$ exists that implements F_f with IT security in presence of a passive adversary in the authenticated channels model. $\Diamond$

Note that by Lemma 2.8.3 we have $\mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}} = \mathcal{F}_{\mathsf{pas}}^{\mathsf{bc}}$ where $\mathcal{F}_{\mathsf{pas}}^{\mathsf{bc}}$ denotes the functions computable by public discussion in the setting above. Hence we may, for the sake of simplicity, assume an authenticated BC channel instead of authenticated channels as the sole underlying resource in the following discussion.

An important subset $\mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$ is the set $\mathcal{F}_{\mathsf{loc}}$ of locally computable n-party functions.

Definition 3.2.2 ($\mathcal{F}_{\mathsf{loc}}$: Locally Computable Functions)**.** A function $f \in \mathcal{F}$ is called *locally computable* ($f \in \mathcal{F}_{\mathsf{loc}}$) if each party P_i can compute its function value $y_i = f_i(x_1, \ldots, x_n)$ locally, without interacting with a resource or another party. $\Diamond$

Obviously, for f to be locally computable, f_i cannot depend on the inputs of parties other than P_i:

Lemma 3.2.3 (Characterization of $\mathcal{F}_{\mathsf{loc}}$)**.** *A function $f \in \mathcal{F}$ is locally computable ($f \in \mathcal{F}_{\mathsf{loc}}$) if and only if for every party $\mathsf{P}_i \in \mathcal{P}$ and every input $x_i \in \mathcal{X}_i$ the restriction $f_i \mid_{\mathcal{X}_1 \times \ldots \times \mathcal{X}_{i-1} \times \{x_i\} \times \mathcal{X}_{i+1} \times \ldots \times \mathcal{X}_n}$ of f is constant.*

Towards a characterization of $\mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$ we give a combinatorial definition of a set $\mathcal{F}'_{\mathsf{pas}}$ of functions that we call *passively decomposable*. Passive decomposability captures the fact that on any restriction of a function $f \in \mathcal{F}'_{\mathsf{pas}}$ a party P_i can send a message about its input such that no adversarial party P_e can learn anything that is not implied by P_e's own input and function output.

Definition 3.2.4 ($\mathcal{F}'_{\mathsf{pas}}$: Passively Decomposable Functions)**.** A function $f \in \mathcal{F}_n$ is called *passively decomposable*, denoted $f \in \mathcal{F}'_{\mathsf{pas}}$, if for any restriction $f \mid_{\widetilde{\mathcal{X}}_1 \times \ldots \times \widetilde{\mathcal{X}}_n}$ of f to subsets $\widetilde{\mathcal{X}}_j \subseteq \mathcal{X}_j$ ($j \in [n]$) we have:

1. $f \mid_{\widetilde{\mathcal{X}}_1 \times \ldots \times \widetilde{\mathcal{X}}_n}$ is locally computable ($f \mid_{\widetilde{\mathcal{X}}_1 \times \ldots \times \widetilde{\mathcal{X}}_n} \in \mathcal{F}_{\mathsf{loc}}$) or
2. there is a party $\mathsf{P}_i \in \mathcal{P}$ and a partition (K-Cut) of the input domain $\widetilde{\mathcal{X}}_i$ into non-empty sets $\mathcal{X}'_i \dot{\cup} \mathcal{X}''_i = \widetilde{\mathcal{X}}_i$ such that for any corrupted party $\mathsf{P}_e \in \mathcal{P} \setminus \{\mathsf{P}_i\}$ and all choices of adversarial input $x_e \in \widetilde{\mathcal{X}}_e$:

$$f_e(x_e, \widetilde{\mathcal{X}}_{\mathcal{H}'}, \mathcal{X}'_i) \cap f_e(x_e, \widetilde{\mathcal{X}}_{\mathcal{H}'}, \mathcal{X}''_i) = \emptyset,$$

where $\mathcal{H}' := \mathcal{H} \setminus \{\mathsf{P}_i\} = \mathcal{P} \setminus \{\mathsf{P}_e, \mathsf{P}_i\}$.

$\Diamond$

The above definition only demands a party P_i can publish which of the two partitions $\mathcal{X}'_i \dot{\cup} \mathcal{X}''_i = \widetilde{\mathcal{X}}_i$ it input x_i is in without compromising its privacy in presence of adversary sets $\mathcal{A} = \{\mathsf{P}_e\}$ of cardinality $|\mathcal{A}| = 1$. As the following lemma states, this is actually equivalent to quantifying over all sets $\mathcal{A} \subseteq \mathcal{P}$ of corrupted parties.

Lemma 3.2.5 (An Equivalent Characterization of $\mathcal{F}'_{\mathsf{pas}}$)**.** *A function $f \in \mathcal{F}_n$ is passively decomposable ($f \in \mathcal{F}'_{\mathsf{pas}}$) if and only if for any restriction $f \mid_{\widetilde{\mathcal{X}}_1 \times \ldots \times \widetilde{\mathcal{X}}_n}$ of f to subsets $\widetilde{\mathcal{X}}_j \subseteq \mathcal{X}_j$ ($j \in [n]$) we have:*

1. *$f \mid_{\widetilde{\mathcal{X}}_1 \times \ldots \times \widetilde{\mathcal{X}}_n}$ is locally computable ($f \mid_{\widetilde{\mathcal{X}}_1 \times \ldots \times \widetilde{\mathcal{X}}_n} \in \mathcal{F}_{\mathsf{loc}}$) or*

2. *there is a party* $\mathsf{P}_i \in \mathcal{P}$ *and a partition (K-Cut) of the input domain* $\widetilde{\mathcal{X}}_i$ *into non-empty sets* $\mathcal{X}_i' \dot{\cup} \mathcal{X}_i'' = \widetilde{\mathcal{X}}_i$ *such that for all sets of corrupted parties* $\emptyset \neq \mathcal{A} \subseteq \mathcal{P} \setminus \{\mathsf{P}_i\}$ *and all choices of adversarial inputs* $x_{\mathcal{A}} \in \widetilde{\mathcal{X}}_{\mathcal{A}}$*:*

$$f_{\mathcal{A}}(x_{\mathcal{A}}, \widetilde{\mathcal{X}}_{\mathcal{H}'}, \mathcal{X}_i') \cap f_{\mathcal{A}}(x_{\mathcal{A}}, \widetilde{\mathcal{X}}_{\mathcal{H}'}, \mathcal{X}_i'') = \emptyset$$

where $\mathcal{H}' := \mathcal{H} \setminus \{\mathsf{P}_i\} = \mathcal{P} \setminus (\mathcal{A} \cup \{\mathsf{P}_i\})$.

The proof of Lemma 3.2.5 is by induction over the size of the adversary set $\mathcal{A}$. Before we prove Lemma 3.2.5, we graphically illustrate passive decomposability. Fig. 3.1 shows the function table of a restriction of f but only with the function outputs of $f_{\mathcal{A}}$. The input domains of the corrupted parties $\mathcal{A}$ of the party P_i and of the remaining honest parties $\mathcal{H}' = \mathcal{H} \setminus \{\mathsf{P}_i\}$ are shown along the three axes of the figure. The definition says, that for any choice of corrupted players $\mathcal{A}$ and inputs $x_{\mathcal{A}}$ for these corrupted players, all the function outputs of $f_{\mathcal{A}}$ in the densely dotted part have to be different from all the function outputs in the sparsely dotted part. As mentioned earlier, the intuition behind the above definition is that party P_i can now tell all the other parties if its input is in $\mathcal{X}_i'$ or in $\mathcal{X}_i''$. From this information, the other parties cannot learn anything more than they will know anyway, once they obtain their respective function outputs.

Proof of Lemma 3.2.5. We prove the two implications separately. It is easy to see that the conditions of Lemma 3.2.5 imply Definition 3.2.4. Condition (1.) is identical and condition if (2.) holds for any set of corrupted parties $\mathcal{A}$ as in Lemma 3.2.5 then it also holds for sets $\mathcal{A}$ of size $|\mathcal{A}| = 1$ as in Definition 3.2.4.

In order to prove the other direction, that Definition 3.2.4 implies the conditions of Lemma 3.2.5 let f be passively decomposable according to Definition 3.2.4 and consider some restriction $\tilde{f}$ of f. If $\tilde{f}$ is locally computable according to condition (1.) of Definition 3.2.4 we are done because then condition (1.) of Lemma 3.2.5 holds. If condition (2.) of Definition 3.2.4 holds, we show by induction on the size of the corruption set $\mathcal{A}$ that indeed condition (2.) of Lemma 3.2.5 is true:

Base case: If we restrict condition (2.) of Lemma 3.2.5 to $|\mathcal{A}| = 1$, the two characterizations are equivalent.

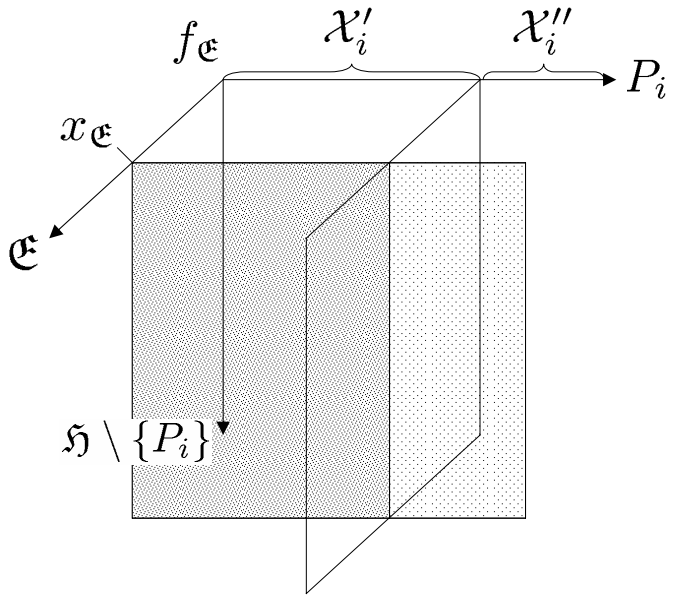


Figure 3.1: Illustration of Passive Decomposability

Induction hypothesis: Now assume that condition (2.) of Lemma 3.2.5 holds for corruption sets $\mathcal{A}$ of size $|\mathcal{A}| < m$ ($m < |\mathcal{P}| - 1$).

Induction step: We show that the two characterizations are equivalent for any set $\mathcal{A}_m$ where $|\mathcal{A}_m| = m$. Let $\tilde{f} = f\,|_{\widetilde{\mathcal{X}}_1 \times \widetilde{\mathcal{X}}_2 \times \ldots \times \widetilde{\mathcal{X}}_n}$ be some restriction of f. Wlog (else rename the parties) consider the following sets of parties with their respective inputs as specified below:

$\mathcal{A}_m = \{\mathsf{P}_1, \ldots, \mathsf{P}_m\}$	$\mathcal{A}_{m-1} = \{\mathsf{P}_1, \ldots, \mathsf{P}_{m-1}\}$	$\widetilde{\mathcal{A}}_m = \{\mathsf{P}_m\}$
$\mathcal{H}_m = \mathcal{P} \setminus \mathcal{A}_m$	$\mathcal{H}_{m-1} = \mathcal{P} \setminus \mathcal{A}_{m-1}$	$\widetilde{\mathcal{H}}_m = \mathcal{P} \setminus \widetilde{\mathcal{A}}_m$
$\mathcal{H}'_m = \mathcal{H}_m \setminus \{\mathsf{P}_i\}$	$\mathcal{H}'_{m-1} = \mathcal{H}_{m-1} \setminus \{\mathsf{P}_i\}$	$\widetilde{\mathcal{H}}'_m = \widetilde{\mathcal{H}}_m \setminus \{\mathsf{P}_i\}$
$x_{\mathcal{A}_m} \in \widetilde{\mathcal{X}}_{\mathcal{A}_m}$	$x_{\mathcal{A}_{m-1}} \in \widetilde{\mathcal{X}}_{\mathcal{A}_{m-1}}$	$x_m \in \widetilde{\mathcal{X}}_{\widetilde{\mathcal{A}}_m} = \mathcal{X}_m$
$x'_{\mathcal{H}'_m} \in \widetilde{\mathcal{X}}_{\mathcal{H}'_m}$	$x'_{\mathcal{H}'_{m-1}} \in \widetilde{\mathcal{X}}_{\mathcal{H}'_{m-1}}$	$x_{\widetilde{\mathcal{H}}'_m} \in \widetilde{\mathcal{X}}_{\widetilde{\mathcal{H}}'_m}$
$x''_{\mathcal{H}'_m} \in \widetilde{\mathcal{X}}_{\mathcal{H}'_m}$	$x''_{\mathcal{H}'_{m-1}} \in \widetilde{\mathcal{X}}_{\mathcal{H}'_{m-1}}$	$x''_{\widetilde{\mathcal{H}}'_m} \in \widetilde{\mathcal{X}}_{\widetilde{\mathcal{H}}'_m}$.

Since the corruption set $|\mathcal{A}_{m-1}| = m-1 < m$ we know by induction hypothesis, that for all $x_{\mathcal{A}_{m-1}}, x'_{\mathcal{H}'_{m-1}}, x''_{\mathcal{H}'_{m-1}}$:

$$f_{\mathcal{A}_{m-1}}(x_{\mathcal{A}_{m-1}}, x'_{\mathcal{H}'_{m-1}}, \mathcal{X}'_i) \cap f_{\mathcal{A}_{m-1}}(x_{\mathcal{A}_{m-1}}, x''_{\mathcal{H}'_{m-1}}, \mathcal{X}''_i) = \emptyset$$

and also by induction hypothesis, since $|\widetilde{\mathcal{A}}_m| = 1 < m$, we have for all $x_m, x'_{\widetilde{\mathcal{H}}'_m}, x''_{\widetilde{\mathcal{H}}'_m}$:

$$f_m(x_m, x'_{\widetilde{\mathcal{H}}'_m}, \mathcal{X}'_i) \cap f_m(x_m, x''_{\widetilde{\mathcal{H}}'_m}, \mathcal{X}''_i) = \emptyset$$

The above two statements tell us that for the parties in $\mathcal{A}_{m-1}$ and the party P_m the intersection is empty for any choice of input values of the parties in $\mathcal{H}'_{m-1}$ and $\widetilde{\mathcal{H}}'_m$ respectively. But now, since $\mathcal{A}_m = \mathcal{A}_{m-1} \cup \widetilde{\mathcal{A}}_m$, $\mathcal{H}'_m \subseteq \mathcal{H}'_{m-1}$ and $\mathcal{H}'_m \subseteq \widetilde{\mathcal{H}}'_m$ this implies that $\forall x_{\mathcal{A}_m}, x'_{\mathcal{H}'_m}, x''_{\mathcal{H}'_m}$:

$$f_{\mathcal{A}_m}(x_{\mathcal{A}_m}, x'_{\mathcal{H}'_m}, \mathcal{X}'_i) \cap f_{\mathcal{A}_m}(x_{\mathcal{A}_m}, x''_{\mathcal{H}'_m}, \mathcal{X}''_i) = \emptyset$$

which concludes the argument. □

We now show that passive decomposability as defined above indeed characterizes the passively computable n-party functions $f \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$:

Theorem 3.2.6. *A function $f \in \mathcal{F}$ is passively computable if and only if it is passively decomposable. In short $\mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}} = \mathcal{F}'_{\mathsf{pas}}$. Furthermore, any function $f \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$ can be computed with PFE security in constantly[14] many rounds.*

The proof of Theorem 3.2.6 can be found below. $\mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}} \subseteq \mathcal{F}'_{\mathsf{pas}}$ is shown by demonstrating that in absence of a K-cut no protocol participant can send a message that bears any information about his input without losing security. The proof of $\mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}} \supseteq \mathcal{F}'_{\mathsf{pas}}$ is constructive in the sense that it inductively describes a passively PFE secure constant round protocol π_f to compute a function $f \in \mathcal{F}'_{\mathsf{pas}}$. The protocol π_f generalizes the approach of [Kus92] to asymmetric n-party functions:

Wlog assume that there is a partition of $\mathcal{X}_i = \mathcal{X}_i^{(1)} \dot{\cup} \mathcal{X}_i^{(2)}$ as described in Definition 3.2.4. The protocol π_f then proceeds as follows: The party P_i determines the message $m_1 \in \{0,1\}$ such that for the input $x_i \in \mathcal{X}_i$ of P_i we have $x_i \in \mathcal{X}_i^{(m_1)}$ and broadcasts m_1. All parties then restrict the function f to $f|_{\mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_i^{(m_1)} \times \ldots \times \mathcal{X}_n}$ and proceed with a partition for

[14] in the security parameter κ

the restricted function in the same fashion. The process is iterated until the parties arrive at a locally computable restriction of f, at which point they can determine the output locally.

We conjecture that the above protocol achieves the optimal round complexity if it is refined to use the finest possible decomposition (according to [Kus92]) of the input domains in every round.

Proof of Theorem 3.2.6. By Lemma 2.8.3 we have $\mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}} = \mathcal{F}_{\mathsf{pas}}^{\mathsf{bc}}$, so it suffices to show that $\mathcal{F}_{\mathsf{pas}}^{\mathsf{bc}} = \mathcal{F}'_{\mathsf{pas}}$. This is simplifies matters because in the public discussion setting we have a global transcript which we can refer to. We then first prove $\mathcal{F}'_{\mathsf{pas}} \subseteq \mathcal{F}_{\mathsf{pas}}^{\mathsf{bc}}$ by inductively describing a passively PFE secure protocol to compute any function in $\mathcal{F}'_{\mathsf{pas}}$. Since the protocol is PFE secure, this implies that it is also IT secure. In the second part of the proof we show that $\mathcal{F}_{\mathsf{pas}}^{\mathsf{bc}} \subseteq \mathcal{F}'_{\mathsf{pas}}$, i.e. that any passively computable function has the structure described in Definition 3.2.4.

$f \in \mathcal{F}'_{\mathsf{pas}} \implies f \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{bc}}$**:** We prove the claim by induction over the size of the input space $|\mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_n|$ of the function f.

Base case: If the function f is locally computable ($f \in \mathcal{F}_{\mathsf{loc}}$), we trivially have $f \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{bc}}$ since all parties can compute the function locally.

Induction hypothesis: We assume that for any $f' \in \mathcal{F}'_{\mathsf{pas}}$ with input space smaller than $|\mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_n|$ we have $f' \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{bc}}$.

Induction step: Since $f \in \mathcal{F}'_{\mathsf{pas}}$ we have two cases: Either $f \in \mathcal{F}_{\mathsf{loc}}$ (this is the base case and we are done) or f has a K-cut according to the definition of $\mathcal{F}'_{\mathsf{pas}}$. This is the case we consider now.

Wlog we can assume (else interchange the parties) that the set $\mathcal{X}_1$ has K-cut $\mathcal{X}_1^{(1)} \dot{\cup} \mathcal{X}_1^{(2)} = \mathcal{X}_1$ (We choose the subset $\widetilde{\mathcal{X}}_1$ from the definition of $\mathcal{F}'_{\mathsf{pas}}$ to be $\mathcal{X}_1$). Now we define

$$f^{(1)} := f\,|_{\mathcal{X}_1^{(1)} \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_n}\,,$$
$$f^{(2)} := f\,|_{\mathcal{X}_1^{(2)} \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_n}\,.$$

By induction hypothesis, $f^{(1)}, f^{(2)} \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{bc}}$ since $f^{(1)}, f^{(2)} \in \mathcal{F}'_{\mathsf{pas}}$ and

$$|\mathcal{X}_1^{(1)} \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_n| < |\mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_n|$$
$$|\mathcal{X}_1^{(2)} \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_n| < |\mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_n|.$$

So there are constant round protocols $\pi^{(1)}$ and $\pi^{(2)}$ that PFE securely implement $\mathsf{F}_{f^{(1)}}$ and $\mathsf{F}_{f^{(2)}}$ and there are corresponding simulators $\mathsf{S}^{(1)}, \mathsf{S}^{(2)} \in \mathfrak{S}_{\mathsf{pas}}$. We construct the protocol π by defining a first protocol round where P_1 sends a message $m_1 \in \{1, 2\}$ to all other parties indicating whether P_1's input x_1 is $x_1 \in \mathcal{X}_1^{(1)}$ or $x_1 \in \mathcal{X}_1^{(2)}$. Subsequently all parties proceed to run protocol $\pi^{(m_1)}$. To prove that the protocol π is PFE secure we provide an appropriate simulator $\mathsf{S} \in \mathfrak{S}_{\mathsf{pas}}$ for the case where an arbitrary subset $\mathcal{A} \subset \mathcal{P}$ of parties is corrupted by a passive adversary $\mathsf{A} \in \mathfrak{A}_{\mathsf{pas}}$. It suffices to look at two cases. Either P_1 is in the set of corrupted parties ($\mathsf{P}_1 \in \mathcal{A}$), or it is not ($\mathsf{P}_1 \notin \mathcal{A}$).

If $\mathsf{P}_1 \notin \mathcal{A}$, we have $\mathcal{A} = \{\mathsf{P}_{e_1}, \ldots, \mathsf{P}_{e_{|\mathcal{A}|}}\}$ where $e_i \in \{2, \ldots, n\}$ and we assume $\mathcal{A} \neq \emptyset$ (If $\mathcal{A} = \emptyset$, no party is corrupted and security is trivially achieved). Now we construct the simulator $\mathsf{S}_{\mathsf{P}_1 \notin \mathcal{A}}$ as follows:

1. $\mathsf{S}_{\mathsf{P}_1 \notin \mathcal{A}}$ fixes the randomness of the adversary A and runs A internally.
2. It feeds the input $x_{\mathsf{A}} = (x_{\mathcal{A}}, x'_{\mathsf{A}})$ of the distinguisher D to the adversary A.[15]
3. It extracts the input $x_{\mathcal{A}} = (x_{e_1}, x_{e_2} \ldots, x_{e_{|\mathcal{A}|}})$ from x_{A} and forwards it to the ideal system F_f.
4. After receiving output $y_{\mathcal{A}} = (y_{e_1}, y_{e_2} \ldots, y_{e_{|\mathcal{A}|}})$ from F_f where $y_{\mathcal{A}} = f_{\mathcal{A}}(x_{\mathcal{A}}, x_{\mathcal{H}'}, x_1)$ and $\mathcal{H}' = \mathcal{H} \setminus \{\mathsf{P}_1\}$, the simulator $\mathsf{S}_{\mathsf{P}_1 \notin \mathcal{A}}$ computes m_1 such that $y_{\mathcal{A}} \in f_{\mathcal{A}}(x_{\mathcal{A}}, x_{\mathcal{H}'}, \mathcal{X}_1^{(m_1)})$.

   Note that this is always uniquely possible since $f \in \mathcal{F}'_{\mathsf{pas}}$ implies (using Lemma 3.2.5) that for any corruption set $\mathcal{A} \subseteq \mathcal{P}$ and for any choice of inputs $x_{\mathcal{A}}$ we have

$$f_{\mathcal{A}}(x_{\mathcal{A}}, \mathcal{X}_{\mathcal{H}'}, \mathcal{X}_1^{(1)}) \cap f_{\mathcal{A}}(x_{\mathcal{A}}, \mathcal{X}_{\mathcal{H}'}, \mathcal{X}_1^{(2)}) = \emptyset.$$

5. Finally $\mathsf{S}_{\mathsf{P}_1 \notin \mathcal{A}}$ inputs m_1 to A and runs $\mathsf{S}^{(m_1)}(\mathsf{A})$ forwarding all inputs and outputs.

[15] x'_{A} stands for any information that the distinguisher D may pass to the adversary A in addition to the actual function inputs $x_{\mathcal{A}}$ that the passive adversary A must use.

The correctness of the the simulation stems from the fact that the simulator $\mathsf{S}_{\mathsf{P}_1 \notin \mathcal{A}}$ can correctly discern the first message m_1 given the output $y_{\mathcal{A}}$ of the ideal functionality F_f. After simulating this message, the remainder of the simulation can be delegated to the subsimulator $\mathsf{S}^{(m_1)}$ that is guaranteed by our induction hypothesis.

It remains to look at the case where $\mathsf{P}_1 \in \mathcal{A}$. We then have $\mathcal{A} = \{\mathsf{P}_1, \mathsf{P}_{e_1}, \ldots, \mathsf{P}_{e_{|\mathcal{A}\setminus\{1\}|}}\}$ where $e_i \in \{2, \ldots, n\}$. The simulator $\mathsf{S}_{\mathsf{P}_1 \in \mathcal{A}}$ can be constructed as follows:

1. Simulator $\mathsf{S}_{\mathsf{P}_1 \in \mathcal{A}}$ fixes the randomness of the adversary A.
2. It feeds the input x_{A} of the distinguisher D to the adversary A and runs it until it outputs the message m_1.
3. $\mathsf{S}_{\mathsf{P}_1 \in \mathcal{A}}$ then runs the simulator $\mathsf{S}^{(m_1)}$ simply forwarding all inputs and outputs.

The simulation is correct since simulator $\mathsf{S}^{(m_1)}$ (which is given by induction hypothesis) now correctly simulates the execution with the honest parties that execute $\pi^{(m_1)}$.

This completes the first part of the proof.

$f \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{bc}} \implies f \in \mathcal{F}'_{\mathsf{pas}}$**:** We give a proof by contradiction. Assume a counterexample $f \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{bc}}$ that is minimal in the size of the input space $|\mathcal{X}_1 \times \ldots \times \mathcal{X}_n|$, such that $f \notin \mathcal{F}'_{\mathsf{pas}}$. Our goal is to show that such a counterexample does not exist.

Since $f \notin \mathcal{F}'_{\mathsf{pas}}$, there is a restriction $\tilde{f} := f\,|_{\widetilde{\mathcal{X}}_1 \times \ldots \times \widetilde{\mathcal{X}}_n}$ of f such that

1. $\tilde{f}$ is **not** locally computable ($\tilde{f} \notin \mathcal{F}_{\mathsf{loc}}$) **and**
2. $\forall\, i : \forall$ partitions of $\widetilde{\mathcal{X}}_i$ into non-empty sets $\mathcal{X}'_i \dot{\cup} \mathcal{X}''_i = \widetilde{\mathcal{X}}_i$
   $\exists\, \mathsf{P}_e \in \mathcal{P} \setminus \{\mathsf{P}_i\}$ (where we define $\mathcal{A} = \{\mathsf{P}_e\}$ and thus $|\mathcal{A}| = 1$) and
   $\exists\, x_e \in \widetilde{\mathcal{X}}_e$ such that:

$$\tilde{f}_e(x_e, \widetilde{\mathcal{X}}_{\mathcal{H}'}, \mathcal{X}'_i) \cap \tilde{f}_e(x_e, \widetilde{\mathcal{X}}_{\mathcal{H}'}, \mathcal{X}''_i) \neq \emptyset$$

   where $\mathcal{H} = \mathcal{P} \setminus \mathcal{A}$ and $\mathcal{H}' := \mathcal{H} \setminus \{\mathsf{P}_i\}$.

Claim 3.2.7. *Due to the minimality of f, the above two conditions (1.) and (2.) hold for the function $\tilde{f} = f$ itself, i.e. f is not locally computable and does not have a K-Cut.*

To see this, we observe that, as $f \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{bc}}$, so is any restriction $\tilde{f} := f\,|_{\widetilde{\mathcal{X}}_1 \times \ldots \times \widetilde{\mathcal{X}}_n}$ of f to subsets $\widetilde{\mathcal{X}}_1 \subseteq \mathcal{X}_1, \ldots, \widetilde{\mathcal{X}}_n \subseteq \mathcal{X}_n$:

$$f \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{bc}} \implies \tilde{f} \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{bc}} \tag{3.1}$$

The exactly same protocol and simulator as for function f can be used for the restriction $\tilde{f}$, merely restricting the input domain. Now assume conditions (1.) and (2.) both hold for a proper restriction $\tilde{f}$ of function f. This means that $\tilde{f}$ is not locally computable and does not have a K-Cut. Therefore we know $\tilde{f} \notin \mathcal{F}'_{\mathsf{pas}}$, which, using Equation (3.1) and the fact that $\tilde{f}$ is a proper restriction, contradicts the minimality of f. Therefore we know that (1.) and (2.) do not both hold in any proper restriction of f and thus the function $f = f\,|_{\mathcal{X}_1 \times \ldots \times \mathcal{X}_n}$ must itself satisfy both conditions (1.) and (2.). This proves Claim 3.2.7.

Now as $f \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{bc}}$ we have a constant round passively IT secure protocol π computing f. Denote the number of rounds of protocol π by ℓ and let x_i denote the input and c_i the random coin tosses of party P_i. We can then define the set of protocol transcripts

$$\begin{aligned}
\Pi := \pi(\mathcal{X}_1, \ldots, \mathcal{X}_n) = \{t = m_1, \ldots, m_\ell \mid \\
\exists\, x_1, c_1, \ldots, x_n, c_n \;:\; t = \pi(x_1, c_1, \ldots, x_n, c_n)\} \\
\Pi_r := \pi(\mathcal{X}_1, \ldots, \mathcal{X}_n)|_r = \{t = m_1, \ldots, m_r \mid \\
\exists\, m_{r+1}, \ldots, m_\ell \;:\; t, m_{r+1}, \ldots, m_\ell \in \Pi\} \\
= \{t = m_1, \ldots, m_r \mid \\
\exists\, x_1, c_1, \ldots, x_n, c_n \;:\; t = \pi(x_1, c_1, \ldots, x_n, c_n)|_r\}.
\end{aligned}$$

We define random variables $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_n$ for the input and output of the parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ respectively. For a set $M \subseteq \mathcal{P}$ we define X_M and Y_M to be the random variable for the inputs and outputs of the parties in M. Then we let $T \in \Pi$ denote the random variable for the transcript, and $T_r \in \Pi_r$ the random variable on transcript prefixes of length r. We name the inputs $\mathcal{X}_l = \{x_l^{(1)}, \ldots, x_l^{(|\mathcal{X}_l|)}\}$ for party $\mathsf{P}_l \in \mathcal{P}$ and similarly $\mathcal{X}_M = \{x_M^{(1)}, \ldots, x_M^{(|\mathcal{X}_M|)}\}$ for a subset M of the parties $\mathcal{P}$. We first show that for any $x_1^{(i_1)} \in \mathcal{X}_1, \ldots, x_n^{(i_n)} \in \mathcal{X}_n$ the statistical distance

$$\begin{aligned}
\Delta(\Pr_{T|X_1,\ldots,X_n}(\cdot, x_1^{(1)}, \ldots, x_n^{(1)}), \\
\Pr_{T|X_1,\ldots,X_n}(\cdot, x_1^{(i_1)}, \ldots, x_n^{(i_n)})) < \mathsf{negl}
\end{aligned} \tag{3.2}$$

of these two distributions on the transcripts $t \in \Pi$ is negligible in the security parameter κ. We proceed by induction over the number of protocol rounds ℓ and show that for any round $r \leq \ell$

$$\delta_r := \max_{i_1,\ldots,i_n} \Delta(\Pr_{T_r|X_1,\ldots,X_n}(\cdot, x_1^{(1)}, \ldots, x_n^{(1)}), \Pr_{T_r|X_1,\ldots,X_n}(\cdot, x_1^{(i_1)}, \ldots, x_n^{(i_n)})) < \mathsf{negl}. \tag{3.3}$$

Base case: For r=0 we have $\Pi_0 = \{\epsilon\}$ where ϵ is the empty transcript. Thus $\delta_0 = 0$.

Induction hypothesis: We assume $\delta_{r-1} < \mathsf{negl}$.

Induction step: Let $t_r = (m_1, \ldots, m_r) \in \Pi_r$ and define $t_{r-1} = (m_1, \ldots, m_{r-1}) \in \Pi_{r-1}$. Wlog (else rename the parties) the message in round r is sent from party P_1 to all other parties.

Recall the formalization of $f \notin \mathcal{F}'_{\mathsf{pas}}$ from the beginning of the proof. As we have seen in Claim 3.2.7, function f does not have a K-Cut. Using this, we can impose an ordering on the elements of input set $\mathcal{X}_1$ such that for any input $x_1^{(j)}$ ($1 < j \leq |\mathcal{X}_1|$) there is another input $x_1^{(k)}$ ($1 \leq k < j$) and a corruption set $\mathcal{A} = \{\mathsf{P}_e\}$ ($\mathsf{P}_e \in \{\mathsf{P}_2, \ldots, \mathsf{P}_n\}$) and an input $x_e^{(jk)} \in \mathcal{X}_{\mathcal{A}} = \mathcal{X}_e$ and inputs $\bar{x}_{\mathcal{H}'}^{(jk)}, \bar{\bar{x}}_{\mathcal{H}'}^{(jk)} \in \mathcal{X}_{\mathcal{H}'}$ such that:

$$f_e(x_e^{(jk)}, \bar{x}_{\mathcal{H}'}^{(jk)}, x_1^{(k)}) = f_e(x_e^{(jk)}, \bar{\bar{x}}_{\mathcal{H}'}^{(jk)}, x_1^{(j)}) \tag{3.4}$$

where we define $\mathcal{H} = \mathcal{P} \setminus \mathcal{A}$ and $\mathcal{H}' = \mathcal{H} \setminus \{\mathsf{P}_1\}$.

Now, by the security of the passively IT secure protocol π we must have for any round r that

$$\Delta(\Pr_{T_r|X_e,X_{\mathcal{H}'},X_1}(\cdot, x_e^{(jk)}, \bar{x}_{\mathcal{H}'}^{(jk)}, x_1^{(k)}), \Pr_{T_r|X_e,X_{\mathcal{H}'},X_1}(\cdot, x_e^{(jk)}, \bar{\bar{x}}_{\mathcal{H}'}^{(jk)}, x_1^{(j)})) < \mathsf{negl} \tag{3.5}$$

This is because a simulator S_e for the corrupted party P_e sees exactly the same (see (3.4)) for the above inputs and is thus unable to emulate two *non-negligibly different* distributions of transcripts.

By definition of statistical distance we have:

$$\begin{aligned}
&\Delta(\Pr_{T_r|X_e,X_{\mathcal{H}'},X_1}(\cdot, x_e^{(1)}, x_{\mathcal{H}'}^{(1)}, x_1^{(k)}), \Pr_{T_r|X_e,X_{\mathcal{H}'},X_1}(\cdot, x_e^{(jk)}, \bar{x}_{\mathcal{H}'}^{(jk)}, x_1^{(k)})) \\
&= \sum_{t_r \in \Pi_r} |\Pr_{T_r|X_e,X_{\mathcal{H}'},X_1}(t_r, x_e^{(1)}, x_{\mathcal{H}'}^{(1)}, x_1^{(k)}) \\
&\qquad\qquad - \Pr_{T_r|X_e,X_{\mathcal{H}'},X_1}(t_r, x_e^{(jk)}, \bar{x}_{\mathcal{H}'}^{(jk)}, x_1^{(k)})|
\end{aligned}$$

We can then factor out the conditional distribution of the message m_r and pull it out of the sum. As message m_r is sent by party P_1 it is conditioned on the transcript so far independent of the inputs of other parties:

$$\begin{aligned}
&= \sum_{t_r \in \Pi_r} |\Pr_{M_r|T_{r-1},X_1}(m_r, t_{r-1}, x_1^{(k)}) \\
&\qquad\qquad \cdot \Pr_{T_{r-1}|X_e,X_{\mathcal{H}'},X_1}(t_{r-1}, x_e^{(1)}, x_{\mathcal{H}'}^{(1)}, x_1^{(k)}) \\
&\quad - \Pr_{M_r|T_{r-1},X_1}(m_r, t_{r-1}, x_1^{(k)}) \\
&\qquad\qquad \cdot \Pr_{T_{r-1}|X_e,X_{\mathcal{H}'},X_1}(t_{r-1}, x_e^{(jk)}, \bar{x}_{\mathcal{H}'}^{(jk)}, x_1^{(k)})| \\
&= \sum_{t_{r-1} \in \Pi_{r-1}} \underbrace{\left(\sum_{m_r} \Pr_{M_r|T_{r-1},X_1}(m_r, t_{r-1}, x_1^{(k)}) \right)}_{=1} \\
&\quad |\Pr_{T_{r-1}|X_e,X_{\mathcal{H}'},X_1}(t_{r-1}, x_e^{(1)}, x_{\mathcal{H}'}^{(1)}, x_1^{(k)}) \\
&\qquad\qquad - \Pr_{T_{r-1}|X_e,X_{\mathcal{H}'},X_1}(t_{r-1}, x_e^{(jk)}, \bar{x}_{\mathcal{H}'}^{(jk)}, x_1^{(k)})|
\end{aligned}$$

We simplify, apply the triangle inequality and the induction hypothesis and find:

$$\begin{aligned}
&= \Delta(\Pr_{T_{r-1}|X_e,X_{\mathcal{H}'},X_1}(t_{r-1}, x_e^{(1)}, x_{\mathcal{H}'}^{(1)}, x_1^{(k)}), \\
&\qquad\qquad \Pr_{T_{r-1}|X_e,X_{\mathcal{H}'},X_1}(t_{r-1}, x_e^{(jk)}, \bar{x}_{\mathcal{H}'}^{(jk)}, x_1^{(k)})) \\
&\leq \Delta(\Pr_{T_{r-1}|X_e,X_{\mathcal{H}'},X_1}(t_{r-1}, x_e^{(1)}, x_{\mathcal{H}'}^{(1)}, x_1^{(1)}), \\
&\qquad\qquad \Pr_{T_{r-1}|X_e,X_{\mathcal{H}'},X_1}(t_{r-1}, x_e^{(1)}, x_{\mathcal{H}'}^{(1)}, x_1^{(k)})) \\
&\quad + \Delta(\Pr_{T_{r-1}|X_e,X_{\mathcal{H}'},X_1}(t_{r-1}, x_e^{(1)}, x_{\mathcal{H}'}^{(1)}, x_1^{(1)}), \\
&\qquad\qquad \Pr_{T_{r-1}|X_e,X_{\mathcal{H}'},X_1}(t_{r-1}, x_e^{(jk)}, \bar{x}_{\mathcal{H}'}^{(jk)}, x_1^{(k)})) \\
&= 2 \cdot \delta_{r-1} < \mathsf{negl}, \qquad (3.6)
\end{aligned}$$

Now let $\tilde{x}_e \in \mathcal{X}_e$ and $\tilde{x}_{\mathcal{H}'} \in \mathcal{X}_{\mathcal{H}'}$ be arbitrary inputs. In the same way as

we derived Inequality (3.5) we find:

$$\begin{aligned}\Delta(&\mathrm{Pr}_{T_r|X_e,X_{\mathcal{H}'},X_1}(\cdot, x_e^{(jk)}, \bar{\bar{x}}_{\mathcal{H}'}^{(jk)}, x_1^{(j)}),\\ &\qquad\qquad \mathrm{Pr}_{T_r|X_e,X_{\mathcal{H}'},X_1}(\cdot, \tilde{x}_e, \tilde{x}_{\mathcal{H}'}, x_1^{(j)}))\\ &\leq 2 \cdot \delta_{r-1} < \mathsf{negl},\end{aligned} \tag{3.7}$$

so we find, using Inequalities (3.5), (3.6) and (3.7), that

$$\begin{aligned}&\Delta(\mathrm{Pr}_{T_r|X_e,X_{\mathcal{H}'},X_1}(\cdot, x_e^{(1)}, x_{\mathcal{H}'}^{(1)}, x_1^{(k)}), \mathrm{Pr}_{T_r|X_e,X_{\mathcal{H}'},X_1}(\cdot, \tilde{x}_e, \tilde{x}_{\mathcal{H}'}, x_1^{(j)}))\\ &\quad\leq \Delta(\mathrm{Pr}_{T_r|X_e,X_{\mathcal{H}'},X_1}(\cdot, x_e^{(1)}, x_{\mathcal{H}'}^{(1)}, x_1^{(k)}),\\ &\qquad\qquad \mathrm{Pr}_{T_r|X_e,X_{\mathcal{H}'},X_1}(\cdot, x_e^{(jk)}, \bar{x}_{\mathcal{H}'}^{(jk)}, x_1^{(k)}))\\ &\qquad + \Delta(\mathrm{Pr}_{T_r|X_e,X_{\mathcal{H}'},X_1}(\cdot, x_e^{(jk)}, \bar{x}_{\mathcal{H}'}^{(jk)}, x_1^{(k)}),\\ &\qquad\qquad \mathrm{Pr}_{T_r|X_e,X_{\mathcal{H}'},X_1}(\cdot, x_e^{(jk)}, \bar{\bar{x}}_{\mathcal{H}'}^{(jk)}, x_1^{(j)}))\\ &\qquad + \Delta(\mathrm{Pr}_{T_r|X_e,X_{\mathcal{H}'},X_1}(\cdot, x_e^{(jk)}, \bar{\bar{x}}_{\mathcal{H}'}^{(jk)}, x_1^{(j)}),\\ &\qquad\qquad \mathrm{Pr}_{T_r|X_e,X_{\mathcal{H}'},X_1}(\cdot, \tilde{x}_e, \tilde{x}_{\mathcal{H}'}, x_1^{(j)}))\\ &\quad\leq \mathsf{negl} + \mathsf{negl} + \mathsf{negl}\end{aligned} \tag{3.8}$$

This is not yet what we claimed in (3.3) because we would like to obtain the above result (3.8) for $k = 1$. We can first observe that the choice of $\tilde{x}_e$ and $\tilde{x}_{\mathcal{H}'}$ and j is arbitrary for the above argument, but $\mathcal{A} = \{\mathsf{P}_e\}$ is fixed for given j. To finally obtain what we claimed in (3.3) we use induction on j. The idea is to apply the result (3.8) step by step:

- To get from j to $k_1 < j$ there exists a set of corrupted parties $\mathcal{A}_1 = \mathsf{P}_{e_1}$ such that the statistical distance in Inequality (3.8) is negligible.
- To get from k_1 to $k_2 < k_1$ there exists a set of corrupted parties $\mathcal{A}_2 = \mathsf{P}_{e_2}$ such that the statistical distance in Inequality (3.8) is negligible.
- ...
- To get from k_i to $1 < k_i$ there exists a set of corrupted parties $\mathcal{A}_i = \mathsf{P}_{e_i}$ such that the statistical distance in Inequality (3.8) is negligible.

We claim that the sum of all these negligible distances is again negligible. This only holds if the length i of the induction chains is at most polynomial in the security parameter κ. But this is given as for a given fixed

function f the sizes of the input domains $|\mathcal{X}_1|, \ldots, |\mathcal{X}_n|$ is of course constant.

This concludes our inductive argument and we proved the claim (3.3).

We know by condition (1.) from the beginning of the proof and the minimality of function f that f is not locally computable. Hence wlog there is a party P_i and an input $x_i \in \mathcal{X}_i$ and inputs $x'_{\mathcal{P}'}, x''_{\mathcal{P}'} \in \mathcal{X}_{\mathcal{P}'}$ where $\mathcal{P}' = \mathcal{P} \setminus \{\mathsf{P}_i\}$ such that

$$f_i(x_i, x'_{\mathcal{P}'}) \neq f_i(x_i, x''_{\mathcal{P}'})$$

From Inequality (3.3) and an application of the triangle inequality we obtain for the transcripts associated with these inputs:

$$\Delta(\mathrm{Pr}_{T|X_i,X_{\mathcal{P}'}}(\cdot, x_i, x'_{\mathcal{P}'}), \mathrm{Pr}_{T|X_A,X_B}(\cdot, x_i, x''_{\mathcal{P}'})) < \mathsf{negl}. \tag{3.9}$$

As the protocol output of P_i is independent of the inputs of the other parties $\mathcal{P}'$ given the transcript, we have:

$$\mathrm{Pr}_{Y_i|T,X_i,X_{\mathcal{P}'}} = \mathrm{Pr}_{Y_i|T,X_i} \tag{3.10}$$

And so we find for the output distribution of protocol π under the given inputs:

$$\begin{aligned}
&\Delta(\mathrm{Pr}_{Y_i|X_i,X_{\mathcal{P}'}}(\cdot, x_i, x'_{\mathcal{P}'}), \mathrm{Pr}_{Y_i|X_i,X_{\mathcal{P}'}}(\cdot, x_i, x''_{\mathcal{P}'})) \\
&= \sum_{y_i \in \mathcal{Y}_i} |\mathrm{Pr}_{Y_i|X_i,X_{\mathcal{P}'}}(y_i, x_i, x'_{\mathcal{P}'}) - \mathrm{Pr}_{Y_i|X_i,X_{\mathcal{P}'}}(y_i, x_i, x''_{\mathcal{P}'})|
\end{aligned}$$

Conditioning on the protocol transcript this becomes:

$$\begin{aligned}
&= \sum_{y_i \in \mathcal{Y}_i} |\sum_{t \in \Pi} \mathrm{Pr}_{Y_i,T|X_i,X_{\mathcal{P}'}}(y_i, t, x_i, x'_{\mathcal{P}'}) - \sum_{t \in \Pi} \mathrm{Pr}_{Y_i,T|X_i,X_{\mathcal{P}'}}(y_i, t, x_i, x''_{\mathcal{P}'})| \\
&\leq \sum_{y_i \in \mathcal{Y}_i, t \in \Pi} |\mathrm{Pr}_{Y_i|T,X_i,X_{\mathcal{P}'}}(y_i, t, x_i, x'_{\mathcal{P}'}) \mathrm{Pr}_{T|X_i,X_{\mathcal{P}'}}(t, x_i, x'_{\mathcal{P}'}) \\
&\qquad\qquad - \mathrm{Pr}_{Y_i|T,X_i,X_{\mathcal{P}'}}(y_i, t, x_i, x''_{\mathcal{P}'}) \mathrm{Pr}_{T|X_i,X_{\mathcal{P}'}}(t, x_i, x''_{\mathcal{P}'})|
\end{aligned}$$

Using Equation (3.10) we obtain

$$\begin{aligned}
&\overset{(3.10)}{=} \sum_{y_i \in \mathcal{Y}_i, t \in \Pi} |\mathrm{Pr}_{Y_i|T,X_i}(y_i, t, x_i) \mathrm{Pr}_{T|X_i,X_{\mathcal{P}'}}(t, x_i, x'_{\mathcal{P}'}) \\
&\qquad\qquad - \mathrm{Pr}_{Y_i|T,X_i}(y_i, t, x_i) \mathrm{Pr}_{T|X_i,X_{\mathcal{P}'}}(t, x_i, x''_{\mathcal{P}'})| \\
&= \sum_{y_i \in \mathcal{Y}_i, t \in \Pi} \mathrm{Pr}_{Y_i|T,X_i}(y_i, t, x_i) \\
&\qquad\qquad |\mathrm{Pr}_{T|X_i,X_{\mathcal{P}'}}(t, x_i, x'_{\mathcal{P}'}) - \mathrm{Pr}_{T|X_i,X_{\mathcal{P}'}}(t, x_i, x''_{\mathcal{P}'})|
\end{aligned}$$

Finally we arrive at

$$= \sum_{t\in\Pi} \underbrace{\left(\sum_{y_i\in\mathcal{Y}_i} \Pr_{Y_i|T,X_i}(y_i,t,x_i) \right)}_{=1} \cdot |\Pr_{T|X_i,X_{\mathcal{P}'}}(t,x_i,x'_{\mathcal{P}'}) - \Pr_{T|X_i,X_{\mathcal{P}'}}(t,x_i,x''_{\mathcal{P}'})|$$

$$= \Delta(\Pr_{T|X_i,X_{\mathcal{P}'}}(\cdot,x_i,x'_{\mathcal{P}'}), \Pr_{T|X_i,X_{\mathcal{P}'}}(\cdot,x_i,x''_{\mathcal{P}'})) \overset{(3.9)}{<} \mathsf{negl}. \tag{3.11}$$

This is in obvious contradiction to the correctness of the protocol π, so we must have $f \in \mathcal{F}'_{\mathsf{pas}}$. Hence there is no counterexample and Theorem 3.2.6 is proven. □

3.3 The Class $\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}}$ of Semi-Honestly Computable Functions

Next we characterize the class $\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}}$ of n-party functions that are IT securely computable in the authenticated channels model in presence of a semi-honest adversary. Semi-honest adversaries, like passive adversaries have to run the prescribed protocol for corrupted parties, but in contrast to passive adversaries semi-honest adversaries may substitute the given inputs of corrupted parties with different ones in order to obtain extra information. The results in this chapter will later help us to characterize LT secure functions in a very practical setting.

Definition 3.3.1 ($\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}}$: Semi-Honestly Computable Functions)**.** The class of *semi-honestly computable* functions $\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}}$ consists of the functions $f \in \mathcal{F}$ for which a constant round protocol π exists that implements F_f with IT security in presence of a semi-honest adversary in the authenticated channels model. ◊

Note that by Lemma 2.8.3 we have $\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} = \mathcal{F}_{\mathsf{sh}}^{\mathsf{bc}}$, where $\mathcal{F}_{\mathsf{sh}}^{\mathsf{bc}}$ denotes the functions computable by public discussion in the setting above. Hence we may, for the sake of simplicity, assume an authenticated BC channel instead of authenticated channels as the sole underlying resource in the following discussion.

We intend to characterize the class $\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}}$ combinatorially. To this end we introduce the concept of redundancy-freeness for n-party functions,

generalizing the 2-party definitions of [KMQ08]. For a party P_i, two of its possible inputs x_i and x_i' to f may be completely indistinguishable to the other parties (by their output from f), while the input x_i may yield a more informative output from f for P_i than x_i'. We then say the input x_i yielding more information dominates the input x_i' giving less information. As semi-honest (and active) adversaries can select their inputs, generally with the goal to obtain as much information as possible, the dominated input x_i' giving less information is not useful to a corrupted P_i. Along the same lines an ideal adversary (simulator) can always use the dominating input x_i instead x_i' of for simulation. As such the input x_i' is redundant, irrelevant in terms of security, and we can eliminate any such input x_i' from the function f under consideration. This yields a *redundancy-free version* $\hat{f}$ of f, with new, smaller, dominating input sets.

Definition 3.3.2 (Domination and Redundancy-Freeness)**.** Given an n-party function $f \in \mathcal{F}_n$ we say $x_i \in \mathcal{X}_i$ *dominates* $x_i' \in \mathcal{X}_i$ if and only if

1. for all $x_{\mathcal{P}'} \in \mathcal{X}_{\mathcal{P}'}$: $f_{\mathcal{P}'}(x_i, x_{\mathcal{P}'}) = f_{\mathcal{P}'}(x_i', x_{\mathcal{P}'})$ and
2. for all $x_{\mathcal{P}'}, x'_{\mathcal{P}'} \in \mathcal{X}_{\mathcal{P}'}$:

$$f_i(x_i', x_{\mathcal{P}'}) \neq f_i(x_i', x'_{\mathcal{P}'}) \implies f_i(x_i, x_{\mathcal{P}'}) \neq f_i(x_i, x'_{\mathcal{P}'}),$$

where $\mathcal{P}' = \mathcal{P} \setminus \{\mathsf{P}_i\}$.

We proceed to define a collection $\bar{\mathcal{X}}_j$ of sets of dominating inputs

$$\bar{\mathcal{X}}_j := \{\mathcal{X} \subseteq \mathcal{X}_j \mid \forall x' \in \mathcal{X}_j \exists x \in \mathcal{X} : x \text{ dominates } x'\} \qquad (j \in [n]).$$

We define the *dominating set* $\hat{\mathcal{X}}_j$ as (some) element of minimal cardinality in $\bar{\mathcal{X}}_j$. We then call $\hat{f} := f|_{\hat{\mathcal{X}}_1 \times \ldots \times \hat{\mathcal{X}}_n}$ the *redundancy-free version* of f. Furthermore, for $x_j \in \mathcal{X}_j$ let $\hat{x}_j \in \hat{\mathcal{X}}_j$ be the (unique) element that dominates the input x_j. $\Diamond$

The redundancy-free version $\hat{f}$ of f is uniquely defined up to a renaming of input and output symbols (also see Section 3.4 or Section 3.7). Domination is a reflexive and transitive relation. Furthermore it is antisymmetric up to renaming of input and output symbols. Two different dominating sets $\hat{\mathcal{X}}_i$ and $\hat{\mathcal{X}}_i'$ which by definition are sets of maximal elements under the domination relation are hence equal up to renaming of input and output symbols.

Since corrupted parties can cooperate to choose their inputs to obtain as much information as possible, it is important to note that the above Definition 3.3.2 generalizes to the combined input of the corrupted parties $\mathcal{A}$ as stated in Lemma 3.3.3 below: If each corrupted party P_{e_j} chooses an input x_{e_j} dominating input x'_{e_j}, then the combined adversarial input $x_\mathcal{A}$ actually dominates $x'_\mathcal{A}$.

Lemma 3.3.3. *Let $\mathcal{A} \subseteq \mathcal{P}$ be a set of corrupted parties and let $x_\mathcal{A} = (x_{e_1}, \ldots, x_{e_{|\mathcal{A}|}}) \in \mathcal{X}_\mathcal{A}$, $x'_\mathcal{A} = (x'_{e_1}, \ldots, x'_{e_{|\mathcal{A}|}}) \in \mathcal{X}_\mathcal{A}$ be inputs such that each x_{e_j} dominates x'_{e_j} ($j \in [|\mathcal{A}|]$). Then we have:*

1. *for all $x_\mathcal{H} \in \mathcal{X}_\mathcal{H}$: $f_\mathcal{H}(x_\mathcal{A}, x_\mathcal{H}) = f_\mathcal{H}(x'_\mathcal{A}, x_\mathcal{H})$ and*
2. *for all $x_\mathcal{H}, x'_\mathcal{H} \in \mathcal{X}_\mathcal{H}$:*

$$f_\mathcal{A}(x'_\mathcal{A}, x_\mathcal{H}) \neq f_\mathcal{A}(x'_\mathcal{A}, x'_\mathcal{H}) \implies f_\mathcal{A}(x_\mathcal{A}, x_\mathcal{H}) \neq f_\mathcal{A}(x_\mathcal{A}, x'_\mathcal{H}).$$

Again we say that $x_\mathcal{A}$ dominates $x'_\mathcal{A}$.

The proof of Lemma 3.3.3 is by induction on the number $|\mathcal{A}|$ of corrupted parties:

Proof of Lemma 3.3.3. For any inputs $x_\mathcal{A} = (x_{e_1}, \ldots, x_{e_{|\mathcal{A}|}})$ and $x'_\mathcal{A} = (x'_{e_1}, \ldots, x'_{e_{|\mathcal{A}|}})$ where each x_{e_j} dominates x'_{e_j} ($j \in [|\mathcal{A}|]$), we have to show that:

1. for all $x_\mathcal{H} \in \mathcal{X}_\mathcal{H}$: $f_\mathcal{H}(x_\mathcal{A}, x_\mathcal{H}) = f_\mathcal{H}(x'_\mathcal{A}, x_\mathcal{H})$ and
2. for all $x_\mathcal{H}, x'_\mathcal{H} \in \mathcal{X}_\mathcal{H}$:

$$f_\mathcal{A}(x'_\mathcal{A}, x_\mathcal{H}) \neq f_\mathcal{A}(x'_\mathcal{A}, x'_\mathcal{H}) \implies f_\mathcal{A}(x_\mathcal{A}, x_\mathcal{H}) \neq f_\mathcal{A}(x_\mathcal{A}, x'_\mathcal{H}).$$

Each x_{e_j} dominates x'_{e_j}, so by definition of domination we have the following:

(i) for all $x_{\mathcal{P}'} \in \mathcal{X}_{\mathcal{P}'}$: $f_{\mathcal{P}'}(x_{e_j}, x_{\mathcal{P}'}) = f_{\mathcal{P}'}(x'_{e_j}, x_{\mathcal{P}'})$ and

(ii) for all $x_{\mathcal{P}'}, x'_{\mathcal{P}'} \in \mathcal{X}_{\mathcal{P}'}$:

$$f_{e_j}(x'_{e_j}, x_{\mathcal{P}'}) \neq f_{e_j}(x'_{e_j}, x'_{\mathcal{P}'}) \implies f_{e_j}(x_{e_j}, x_{\mathcal{P}'}) \neq f_{e_j}(x_{e_j}, x'_{\mathcal{P}'}),$$

where $\mathcal{P}' = \mathcal{P} \setminus \{\mathsf{P}_{e_j}\}$.

To prove claim (1.) of the lemma, we use induction on the number $|\mathcal{A}|$ of corrupted parties.

Base case: We have $|\mathcal{A}| = 1$ and the claim is exactly what is given by (i) above.

Induction hypothesis: Assume the claim (1.) holds for $|\mathcal{A}| = m - 1$.

Induction step: We show that the claim holds for $|\mathcal{A}| = m$. By induction hypothesis we have

$$\forall x_{\mathcal{H}} \in \mathcal{X}_{\mathcal{H}} : f_{\mathcal{H}}(x_{\mathcal{A}}, x_{\mathcal{H}}) = f_{\mathcal{H}}(x'_{\mathcal{A}}, x_{\mathcal{H}})$$

for $x_{\mathcal{A}} = (x_{e_1}, x_{e_2}, \ldots, x_{e_{m-1}})$. Now we add the corrupted party P_{e_m} and let $\mathcal{H}' = \mathcal{H} \setminus \{\mathsf{P}_{e_m}\}$. From the above we know in particular:

$$\forall x_{\mathcal{H}'} \in \mathcal{X}_{\mathcal{H}'} : f_{\mathcal{H}}(x_{\mathcal{A}}, x'_{e_m}, x_{\mathcal{H}'}) = f_{\mathcal{H}}(x'_{\mathcal{A}}, x'_{e_m}, x_{\mathcal{H}'}) \tag{3.12}$$

But now by (i) for x_{e_m} we can conclude that

$$\begin{aligned} \forall x_{\mathcal{H}'} \in \mathcal{X}_{\mathcal{H}'} : f_{\mathcal{H}'}(x_{\mathcal{A}}, x_{e_m}, x_{\mathcal{H}'}) &\overset{\text{(i)}}{=} f_{\mathcal{H}'}(x_{\mathcal{A}}, x'_{e_m}, x_{\mathcal{H}'}) \\ &\overset{(3.12)}{=} f_{\mathcal{H}'}(x'_{\mathcal{A}}, x'_{e_m}, x_{\mathcal{H}'}) \end{aligned}$$

which is exactly what we wanted to show.

To prove claim (2.), we again use induction on the number $|\mathcal{A}|$ of corrupted parties.

Base case: We have $|\mathcal{A}| = 1$ and the claim is exactly what is given by (ii) above.

Induction hypothesis: Assume the claim (2.) holds for $|\mathcal{A}| = m - 1$.

Induction step: We show that the claim holds for $|\mathcal{A}| = m$. By our induction hypothesis we have

$$\begin{aligned} &\forall x_{\mathcal{H}}, x'_{\mathcal{H}} \in \mathcal{X}_{\mathcal{H}} : \\ &\quad f_{\mathcal{A}}(x'_{\mathcal{A}}, x_{\mathcal{H}}) \neq f_{\mathcal{A}}(x'_{\mathcal{A}}, x'_{\mathcal{H}}) \implies f_{\mathcal{A}}(x_{\mathcal{A}}, x_{\mathcal{H}}) \neq f_{\mathcal{A}}(x_{\mathcal{A}}, x'_{\mathcal{H}}) \end{aligned} \tag{3.13}$$

for $x_{\mathcal{A}} = (x_{e_1}, x_{e_2}, \ldots, x_{e_{m-1}})$. Now we add the corrupted party P_{e_m} and let $\mathcal{H}' = \mathcal{H} \setminus \{\mathsf{P}_{e_m}\}$. Then we find the following:

$$\begin{aligned} &\forall x_{\mathcal{H}'}, x'_{\mathcal{H}'} \in \mathcal{X}_{\mathcal{H}'} : \\ &\qquad f_{\mathcal{A}}(x'_{\mathcal{A}}, x'_{e_m}, x_{\mathcal{H}'}) \neq f_{\mathcal{A}}(x'_{\mathcal{A}}, x'_{e_m}, x'_{\mathcal{H}'}) \\ &\implies f_{\mathcal{A}}(x_{\mathcal{A}}, x'_{e_m}, x_{\mathcal{H}'}) \neq f_{\mathcal{A}}(x_{\mathcal{A}}, x'_{e_m}, x'_{\mathcal{H}'}) \\ &\implies f_{\mathcal{A}}(x_{\mathcal{A}}, x_{e_m}, x_{\mathcal{H}'}) \neq f_{\mathcal{A}}(x_{\mathcal{A}}, x_{e_m}, x'_{\mathcal{H}'}) \end{aligned} \tag{3.14}$$

We get the first implication from Equation (3.13) and the second implication becomes clear from the following observation: By (i) for x'_{e_m} we have:

$$\begin{aligned} f_{\mathcal{A}}(x_{\mathcal{A}}, x'_{e_m}, x_{\mathcal{H}'}) &= f_{\mathcal{A}}(x_{\mathcal{A}}, x_{e_m}, x_{\mathcal{H}'}) \\ f_{\mathcal{A}}(x_{\mathcal{A}}, x'_{e_m}, x'_{\mathcal{H}'}) &= f_{\mathcal{A}}(x_{\mathcal{A}}, x_{e_m}, x'_{\mathcal{H}'}) \end{aligned}$$

Now on the other hand we find:

$$\begin{aligned} &\forall x_{\mathcal{H}'}, x'_{\mathcal{H}'} \in \mathcal{X}_{\mathcal{H}'} : \\ &\qquad f_{e_m}(x'_{\mathcal{A}}, x'_{e_m}, x_{\mathcal{H}'}) \neq f_{e_m}(x'_{\mathcal{A}}, x'_{e_m}, x'_{\mathcal{H}'}) \\ &\implies f_{e_m}(x'_{\mathcal{A}}, x_{e_m}, x_{\mathcal{H}'}) \neq f_{e_m}(x'_{\mathcal{A}}, x_{e_m}, x'_{\mathcal{H}'}) \\ &\implies f_{e_m}(x_{\mathcal{A}}, x_{e_m}, x_{\mathcal{H}'}) \neq f_{e_m}(x_{\mathcal{A}}, x_{e_m}, x'_{\mathcal{H}'}) \end{aligned} \tag{3.15}$$

Where the first implication is a special case of property (ii) above for x_{e_m} and the second implication is by property (i) above for $x_{e_1}, \ldots, x_{e_{m-1}}$.

Now we can conclude from Equations (3.14) and (3.15) that for corruption set $\mathcal{A}' = \mathcal{A} \cup \{\mathsf{P}_{e_m}\}$

$$\begin{aligned} &\forall x_{\mathcal{H}'}, x'_{\mathcal{H}'} \in \mathcal{X}_{\mathcal{H}'} : \\ &\quad f_{\mathcal{A}'}(x'_{\mathcal{A}'}, x_{\mathcal{H}'}) \neq f_{\mathcal{A}'}(x'_{\mathcal{A}'}, x'_{\mathcal{H}'}) \implies f_{\mathcal{A}'}(x_{\mathcal{A}'}, x_{\mathcal{H}'}) \neq f_{\mathcal{A}'}(x_{\mathcal{A}'}, x'_{\mathcal{H}'}) \end{aligned}$$

holds and this is what we wanted to prove. □

The following lemma states that the function f and its redundancy free version $\hat{f}$ are mutually locally[16] reducible. This means that it does not matter in terms of security which of the two functions is used and redundant inputs can safely be eliminated.

Lemma 3.3.4. *The functions f and $\hat{f}$ are efficiently and PFE securely mutually locally reducible, even in presence of active adversaries.*

[16] without using any communication resources

The proof of Lemma 3.3.4 is fairly straightforward, by showing how to implement functionality $\mathsf{F}_{\hat{f}}$ when functionality F_f is given and vice versa. The protocol essentially replaces inputs x_i with dominating inputs $\hat{x}_i$.

Proof. We have to show that F_f and $\mathsf{F}_{\hat{f}}$ are mutually locally reducible. We prove each direction separately.

We proceed as follows:

1. We show how to implement $\mathsf{F}_{\hat{f}}$ when F_f is given.
2. We show how to implement F_f when $\mathsf{F}_{\hat{f}}$ is given.

For both cases we show that the implementation is correct and secure.

Let F_f be given. We implement $\mathsf{F}_{\hat{f}}$ using a protocol $\pi_{\hat{f}}$ that simply restricts the input space for F_f. So in this case the real system is $\pi_{\hat{f}} \circ \mathsf{F}_f$ and the ideal functionality is $\mathsf{F}_{\hat{f}}$. Correctness (for the all honest case) is obvious. To prove security we have to provide a efficient simulator $\mathsf{S} \in \mathsf{Poly}$ that interacts with the ideal system $\mathsf{F}_{\hat{f}}$. We assume some nonempty set of parties $\mathcal{A} = \{\mathsf{P}_{e_1}, \ldots, \mathsf{P}_{e_{|\mathcal{A}|}}\} \subseteq \mathcal{P}$ is corrupted. The simulator $\mathsf{S}_{\mathcal{A}}$ simply runs the adversary A until A produces an input $x_{\mathcal{A}} = (x_{e_1}, \ldots, x_{e_{|\mathcal{A}|}})$ intended for functionality F_f. Simulator $\mathsf{S}_{\mathcal{A}}$ then inputs $\hat{x}_{\mathcal{A}} = (\hat{x}_{e_1}, \ldots, \hat{x}_{e_{|\mathcal{A}|}})$ to functionality $\mathsf{F}_{\hat{f}}$ and receives an output $\hat{y}_{\mathcal{A}} = \hat{f}_{\mathcal{A}}(\hat{x}_{\mathcal{A}}, x_{\mathcal{H}})$ where $x_{\mathcal{H}}$ is the input of the honest parties $\mathcal{H} = \mathcal{P} \setminus \mathcal{A}$. Now by Lemma 3.3.3 we have:

1. for all $x_{\mathcal{H}} \in \mathcal{X}_{\mathcal{H}}$: $f_{\mathcal{H}}(\hat{x}_{\mathcal{A}}, x_{\mathcal{H}}) = f_{\mathcal{H}}(x_{\mathcal{A}}, x_{\mathcal{H}})$ and
2. for all $x_{\mathcal{H}}, x'_{\mathcal{H}} \in \mathcal{X}_{\mathcal{H}}$:

$$f_{\mathcal{A}}(x_{\mathcal{A}}, x_{\mathcal{H}}) \neq f_{\mathcal{A}}(x_{\mathcal{A}}, x'_{\mathcal{H}}) \implies f_{\mathcal{A}}(\hat{x}_{\mathcal{A}}, x_{\mathcal{H}}) \neq f_{\mathcal{A}}(\hat{x}_{\mathcal{A}}, x'_{\mathcal{H}}).$$

By point (2.) the simulator $\mathsf{S}_{\mathcal{A}}$ can deduce $y_{\mathcal{A}} = f(x_{\mathcal{A}}, x_{\mathcal{H}})$ from $\hat{y}_{\mathcal{A}}$. The simulator $\mathsf{S}_{\mathcal{A}}$ then feeds $y_{\mathcal{A}}$ to the adversary A and forwards its output y_{A} to the distinguisher D. This clearly results in a perfectly indistinguishable interaction for A and hence in indistinguishable output y_{A} for the distinguisher D. Due to the point (1.) the simulation is also indistinguishable by the outputs of the honest parties $y_{\mathcal{H}}$, thus $\pi_{\hat{f}} \circ \mathsf{F}_f$ indeed perfectly securely implements $\mathsf{F}_{\hat{f}}$ and the simulator $\mathsf{S}_{\mathcal{A}}$ is efficient.

Conversely, let $\mathsf{F}_{\hat{f}}$ be given. We describe the protocol π for implementing functionality F_f from functionality $\mathsf{F}_{\hat{f}}$ by describing party P_i's

protocol π_i for any i. The protocol π_i takes input x_i and inputs $\hat{x}_i$ to $\mathsf{F}_{\hat{f}'}$ receiving $\hat{y}_i = \hat{f}_i(\hat{x}_i, \hat{x}_{\mathcal{P}'})$ in turn (where $\mathcal{P}' = \mathcal{P} \setminus \{\mathsf{P}_i\}$). Now by point (1.) in the definition of $\hat{x}_i$ and $\hat{f}$ we have

$$\hat{y}_i = \hat{f}_i(\hat{x}_i, \hat{x}_{\mathcal{P}'}) = f_i(\hat{x}_i, x_{\mathcal{P}'}).$$

By point (2.) in the definition of $\hat{x}_i$ we then have for all $x_{\mathcal{P}'}, x'_{\mathcal{P}'} \in \mathcal{X}_{\mathcal{P}'}$:

$$f_i(x_i, x_{\mathcal{P}'}) \neq f_i(x_i, x'_{\mathcal{P}'}) \implies f_i(\hat{x}_i, x_{\mathcal{P}'}) \neq f_i(\hat{x}_i, x'_{\mathcal{P}'}).$$

Once again this implies that protocol π_i can recover $y_i = f_i(x_i, x_{\mathcal{P}'})$ from $\hat{y}_i$. Finally, the protocol π_i outputs y_i to the distinguisher. The correctness of the protocol for the all honest case is immediate. For the security proof, we note that in this case the ideal system is F_f and the real system is $\pi_i \circ \mathsf{F}_{\hat{f}}$. Security follows trivially: The simulator $\mathsf{S}_{\mathcal{A}}$ simply restricts the input space of corrupted parties to functionality F_f to $\hat{\mathcal{X}}_{\mathcal{A}}$. □

As PFE security in presence of active adversaries implies ITE security in presence of semi-honest adversaries, we can derive the following simple corollary:

Corollary 3.3.5. *For any function $f \in \mathcal{F}$ we have: $f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} \iff \hat{f} \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}}$.*

An n-party function f is then sh computable if and only if its redundancy-free version $\hat{f}$ is pas computable.

Theorem 3.3.6. *For a function $f \in \mathcal{F}$ we have: $f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} \iff \hat{f} \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$.*

Before we begin with the full proof of Theorem 3.3.6 we provide a short sketch: By Corollary 3.3.5 we know that $f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} \iff \hat{f} \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}}$. Therefore it suffices to show for redundancy-free functions f where $f = \hat{f}$ that we have $f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} \iff f \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$. The implication $f \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}} \implies f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}}$ is then clear by definition. The implication $f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} \implies f \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$ is shown along the lines of the proof of Theorem 3.2.6, demonstrating that $f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} \implies f \in \mathcal{F}'_{\mathsf{pas}}$. The proof exploits the redundancy-freeness of f due to which a (working) simulator in the semi-honest setting cannot actually substitute inputs.

Proof of Theorem 3.3.6. From Corollary 3.3.5 we know that $f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} \iff \hat{f} \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}}$. Therefore it suffices to show for redundancy-free functions where $f = \hat{f}$ that

$$f = \hat{f} \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} \iff f = \hat{f} \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$$

because then we have:

$$f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} \overset{\text{Cor. 3.3.5}}{\iff} \hat{f} \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} \iff \hat{f} \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}.$$

So wlog we assume for the remainder of this proof that the function f is redundancy-free. Now by Lemma 2.8.3 we have $\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} = \mathcal{F}_{\mathsf{sh}}^{\mathsf{bc}}$, so it suffices to show

$$f = \hat{f} \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{bc}} \iff f = \hat{f} \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}. \tag{3.16}$$

This is simplifies matters because in the public discussion setting we have a global transcript which we can refer to. We proceed by showing each implication separately.

$f \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}} \implies f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{bc}}$**:** Since passive security implies semi-honest security (see Section 2.7) this is given.

$f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{bc}} \implies f \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$**:** The argument we give below is very similar to the second part of the proof of Theorem 3.2.6 where we showed that $f \in \mathcal{F}_{\mathsf{pas}}^{\mathsf{bc}} \implies f \in \mathcal{F}'_{\mathsf{pas}}$. But there are two main differences:

(i) Here we do not assume a minimal (in the size of the input space) counterexample, rather we argue over all possible proper restrictions $\tilde{f}$ of function f.

This is necessary as $f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{bc}}$ does not directly imply that a restriction $\tilde{f}$ of f is also in $\mathcal{F}_{\mathsf{sh}}^{\mathsf{bc}}$ and therefore we cannot immediately show an analogue of Claim 3.2.7.

(ii) Since now we are in the semi-honest setting, the simulator is allowed to substitute the inputs provided by the distinguisher for different ones. Therefore, compared to the proof in the passive setting, we have to give an additional argument to prove Inequality (3.20) which corresponds to Inequality (3.5) in the proof for the passive case. At that point we use the fact that $f = \hat{f}$ is redundancy-free to show that the simulator indeed has to forward the inputs provided by the distinguisher.

We give a proof by contradiction. Assume a counterexample $f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{bc}}$ such that $f \notin \mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$. Our goal is to show that such a counterexample does not exist.

Recall that wlog $f = \hat{f}$ is redundancy-free. Since $f \notin \mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$, there is a restriction $\tilde{f} := f\,|_{\widetilde{\mathcal{X}}_1 \times \ldots \times \widetilde{\mathcal{X}}_n}$ of f such that

1. $\tilde{f}$ is **not** locally computable ($\tilde{f} \notin \mathcal{F}_{\mathsf{loc}}$) **and**

2. $\forall\, i : \forall$ partitions of $\widetilde{\mathcal{X}}_i$ into non-empty sets $\mathcal{X}_i' \dot\cup \mathcal{X}_i'' = \widetilde{\mathcal{X}}_i$
   $\exists\, \mathsf{P}_e \in \mathcal{P} \setminus \{\mathsf{P}_i\}$ (where we define $\mathcal{A} = \{\mathsf{P}_e\}$ and thus $|\mathcal{A}| = 1$) and
   $\exists\, x_e \in \widetilde{\mathcal{X}}_e$ such that:

$$\tilde{f}_e(x_e, \widetilde{\mathcal{X}}_{\mathcal{H}'}, \mathcal{X}_i') \cap \tilde{f}_e(x_e, \widetilde{\mathcal{X}}_{\mathcal{H}'}, \mathcal{X}_i'') \neq \emptyset$$

   where $\mathcal{H} = \mathcal{P} \setminus \mathcal{A}$ and $\mathcal{H}' := \mathcal{H} \setminus \{\mathsf{P}_i\}$.

Now as $f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{bc}}$ we have a constant round semi-honestly IT secure protocol π computing f. Denote the number of rounds of protocol π by ℓ and let x_i denote the input and c_i the random coin tosses of party P_i. We can then define the set of protocol transcripts

$$\begin{aligned}
\Pi &:= \pi(\mathcal{X}_1, \ldots, \mathcal{X}_n) = \{t = m_1, \ldots, m_\ell \mid \\
&\qquad \exists\, x_1, c_1, \ldots, x_n, c_n \;:\; t = \pi(x_1, c_1, \ldots, x_n, c_n)\} \\
\Pi_r &:= \pi(\mathcal{X}_1, \ldots, \mathcal{X}_n)|_r = \{t = m_1, \ldots, m_r \mid \\
&\qquad \exists\, m_{r+1}, \ldots, m_\ell \;:\; t, m_{r+1}, \ldots, m_\ell \in \Pi\} \\
&= \{t = m_1, \ldots, m_r \mid \\
&\qquad \exists\, x_1, c_1, \ldots, x_n, c_n \;:\; t = \pi(x_1, c_1, \ldots, x_n, c_n)|_r\}.
\end{aligned}$$

We define random variables $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_n$ for the input and output of the parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ respectively. For a set $M \subseteq \mathcal{P}$ we define X_M and Y_M to be the random variable for the inputs and outputs of the parties in M. Then we let $T \in \Pi$ denote the random variable for the transcript, and $T_r \in \Pi_r$ the random variable on transcript prefixes of length r. Recall that $\tilde{f} = f\,|_{\widetilde{\mathcal{X}}_1 \times \ldots \times \widetilde{\mathcal{X}}_n}$ from the beginning of the proof. We name the inputs $\widetilde{\mathcal{X}}_l = \{x_l^{(1)}, \ldots, x_l^{(|\widetilde{\mathcal{X}}_l|)}\}$ for party $\mathsf{P}_l \in \mathcal{P}$ and similarly $\widetilde{\mathcal{X}}_M = \{x_M^{(1)}, \ldots, x_M^{(|\widetilde{\mathcal{X}}_M|)}\}$ for a subset M of the set of parties $\mathcal{P}$. We first show that for any $x_1^{(i_1)} \in \widetilde{\mathcal{X}}_1, \ldots, x_n^{(i_n)} \in \widetilde{\mathcal{X}}_n$ (from the restricted input space) the statistical distance

$$\begin{aligned}
\Delta(&\mathrm{Pr}_{T|X_1,\ldots,X_n}(\cdot, x_1^{(1)}, \ldots, x_n^{(1)}), \\
&\qquad \mathrm{Pr}_{T|X_1,\ldots,X_n}(\cdot, x_1^{(i_1)}, \ldots, x_n^{(i_n)})) < \mathsf{negl}
\end{aligned} \tag{3.17}$$

of these two distributions on the transcripts $t \in \Pi$ is negligible in the security parameter κ. We proceed by induction over the number of protocol

rounds ℓ and show that for any round $r \leq \ell$

$$\delta_r := \max_{i_1,\ldots,i_n} \Delta(\mathrm{Pr}_{T_r|X_1,\ldots,X_n}(\cdot, x_1^{(1)}, \ldots, x_n^{(1)}), \mathrm{Pr}_{T_r|X_1,\ldots,X_n}(\cdot, x_1^{(i_1)}, \ldots, x_n^{(i_n)})) < \mathsf{negl}. \tag{3.18}$$

Base case: For r=0 we have $\Pi_0 = \{\epsilon\}$ where ϵ is the empty transcript. Thus $\delta_0 = 0$.

Induction hypothesis: We assume $\delta_{r-1} < \mathsf{negl}$.

Induction step: Let $t_r = (m_1, \ldots, m_r) \in \Pi_r$ and consider $t_{r-1} = (m_1, \ldots, m_{r-1}) \in \Pi_{r-1}$. Wlog (else rename the parties) the message in round r is sent from party P_1 to all other parties.

Recall the formalization of $f \notin \mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$ from the beginning of the proof: We know by property (2.) of $\tilde{f}$ that the restricted function $\tilde{f}$ does not have a K-Cut. Using this, we can impose an ordering on the elements of input set $\widetilde{\mathcal{X}}_1$ such that for any input $x_1^{(j)} \in \widetilde{\mathcal{X}}_1$ ($1 < j \leq |\widetilde{\mathcal{X}}_1|$) there is another input $x_1^{(k)} \in \widetilde{\mathcal{X}}_1$ ($1 \leq k < j$) and a corruption set $\mathcal{A} = \{\mathsf{P}_e\}$ ($\mathsf{P}_e \in \{\mathsf{P}_2, \ldots, \mathsf{P}_n\}$) and an input $x_e^{(jk)} \in \widetilde{\mathcal{X}}_{\mathcal{A}} = \widetilde{\mathcal{X}}_e$ and inputs $\bar{x}_{\mathcal{H}'}^{(jk)}, \bar{\bar{x}}_{\mathcal{H}'}^{(jk)} \in \widetilde{\mathcal{X}}_{\mathcal{H}'}$ such that:

$$f_e(x_e^{(jk)}, \bar{x}_{\mathcal{H}'}^{(jk)}, x_1^{(k)}) = f_e(x_e^{(jk)}, \bar{\bar{x}}_{\mathcal{H}'}^{(jk)}, x_1^{(j)}) \tag{3.19}$$

where we define $\mathcal{H} = \mathcal{P} \setminus \mathcal{A}$ and $\mathcal{H}' = \mathcal{H} \setminus \{\mathsf{P}_1\}$.

We proceed by showing that Inequality (3.5) from the proof of Theorem 3.2.6 still holds in the semi-honest case for redundancy free functions f. For any round r we claim:

$$\Delta(\mathrm{Pr}_{T_r|X_e,X_{\mathcal{H}'},X_1}(\cdot, x_e^{(jk)}, \bar{x}_{\mathcal{H}'}^{(jk)}, x_1^{(k)}), \mathrm{Pr}_{T_r|X_e,X_{\mathcal{H}'},X_1}(\cdot, x_e^{(jk)}, \bar{\bar{x}}_{\mathcal{H}'}^{(jk)}, x_1^{(j)})) < \mathsf{negl}. \tag{3.20}$$

In order to prove this claim, we make use of the fact that the protocol π is semi-honestly secure. By the security of π we know that for any set of corrupted parties $\mathcal{A} = \{\mathsf{P}_e\}$ there exists a simulator $\mathsf{S}_e \in \mathfrak{S}_{\mathsf{sh}}$ such that no distinguisher D is able to distinguish the ideal from the real setting.

We observe that if simulator S_e merely forwards the inputs provided by distinguisher D, then Inequality (3.20) holds because by Equation (3.19) simulator S_e has exactly the same view when simulating the transcripts for the two scenarios of Inequality (3.20). But since we consider semi-honest adversaries, the simulator S_e might input a value $\tilde{x}_e \neq x_e^{(jk)}$ to functionality F_f when provided with input $x_e^{(jk)}$ by the distinguisher. Now we show that due to the fact that function f is redundancy-free, a successful simulator S_e must forward the input $x_e^{(jk)}$ provided by the distinguisher D to the ideal functionality F_f. If simulator S_e provides a different input, we exhibit a distinguisher telling the real and ideal settings apart with non-negligible probability, contradicting our assumption that simulator S_e renders the two settings indistinguishable.

Towards a contradiction assume that simulator S_e inputs $\tilde{x}_e \neq x_e^{(jk)}$ to functionality F_f with non-negligible probability. As function f is redundancy-free, we know that input $x_e^{(jk)}$ does not dominate input $\tilde{x}_e$ so by definition of domination there are two possibilities:

$$\exists x_{\mathcal{H}} \in \mathcal{X}_{\mathcal{H}} \ : \ f_{\mathcal{H}}(x_e^{(jk)}, x_{\mathcal{H}}) \neq f_{\mathcal{H}}(\tilde{x}_e, x_{\mathcal{H}}) \qquad \text{or} \tag{3.21}$$

$$\exists x'_{\mathcal{H}}, x''_{\mathcal{H}} \in \mathcal{X}_{\mathcal{H}} \ : \ f_e(x_e^{(jk)}, x'_{\mathcal{H}}) \neq f_e(x_e^{(jk)}, x''_{\mathcal{H}}) \wedge f_e(\tilde{x}_e, x'_{\mathcal{H}}) = f_e(\tilde{x}_e, x''_{\mathcal{H}}). \tag{3.22}$$

If Equation (3.21) holds with non-negligible probability, a distinguisher D can easily distinguish by inputting $x_{\mathcal{H}}$ and comparing the outputs of the honest parties, contradicting our assumption that under simulator S_e, the real and ideal settings are indistinguishable.

So assume Equation (3.22) holds with non-negligible probability. We know that the adversary A runs the protocol π_e because we are in the semi-honest setting. Furthermore, we can assume that adversary A includes the protocol output y_e in its output y_{A}, as a simulator for such an adversary A is easily adapted to work if the protocol output y_e is not included in y_{A}.

Now we construct a distinguisher D that selects the input $\tilde{x}_{\mathcal{H}}$ for the honest parties uniformly at random from $\{x'_{\mathcal{H}}, x''_{\mathcal{H}}\}$. In the real setting we find $y_e = f_e(x_e^{(jk)}, \tilde{x}_{\mathcal{H}})$ with overwhelming probability. But in the ideal setting we find $y_e \neq f_e(x_e^{(jk)}, \tilde{x}_{\mathcal{H}})$ with non-negligible probability because the simulator receives

$$y'_e = f_e(\tilde{x}_e, x'_{\mathcal{H}}) = f_e(\tilde{x}_e, x''_{\mathcal{H}})$$

with non-negligible probability. Then, in the ideal setting y_e is independent of $\tilde{x}_{\mathcal{H}}$ whereas in the real setting it is directly dependent on $\tilde{x}_{\mathcal{H}}$. So the distinguisher D can distinguish the two settings by checking whether $y_e = f_e(x_e^{(jk)}, \tilde{x}_{\mathcal{H}})$. This is a contradiction to the indistinguishability of $\mathsf{S}_e(\mathsf{A}) \circ \mathsf{F}_f$ and $\mathsf{A} \circ \pi$.

This concludes the argument, demonstrating that a successful simulator S_e forwards the input it receives from the distinguisher D with overwhelming probability. By the argument above Inequality (3.20) follows. As we have shown that simulator S_e behaves essentially like a passive simulator $\mathsf{S} \in \mathfrak{S}_{\mathsf{pas}}$, we can proceed to show that $f \in \mathcal{F}'_{\mathsf{pas}}$ exactly as in the proof of Theorem 3.2.6 (for the passive case). By Theorem 3.2.6 we then find $f \in \mathcal{F}^{\mathsf{aut}}_{\mathsf{pas}} = \mathcal{F}'_{\mathsf{pas}}$, which proves Theorem 3.3.6. □

The functions $f^{(5)}$ and $f^{(6)}$ in Fig. 3.2 are examples of *not* sh computable functions taken from [Kus92]. The function $f^{(6)}$ is of particular interest as it is of strictly less cryptographic strength than oblivious transfer. Function $f^{(9)}$ is sh computable: After eliminating the redundant input x_3, the function is pas computable (as indicated by the horizontal and vertical lines).

3.4 Symmetrization

In this section we show that computing a semi-honestly computable function $f \in \mathcal{F}^{\mathsf{aut}}_{\mathsf{sh}}$, is equivalent to computing a specific redundancy free, symmetric function $\mathrm{sym}(f)$. In particular this implies that the function $\mathrm{sym}(f)$ can be computed securely in precisely the same adversarial setting as function f. Our result constitutes a generalization of the Symmetrization-Lemma of [KMQ08] to the n-party setting.

Lemma 3.4.1 (Symmetrization)**.** *For any function $f \in \mathcal{F}^{\mathsf{aut}}_{\mathsf{sh}}$ there is a redundancy free, symmetric function $\mathrm{sym}(f) \in \mathcal{F}$ such that the functions f and $\mathrm{sym}(f)$ are efficiently and PFE securely mutually locally reducible, even in presence of active adversaries.*

Proof. By Lemma 3.3.4 it is sufficient to show Lemma 3.4.1 for redundancy-free functions f where $f = \hat{f}$. So in the following let $f \in \mathcal{F}^{\mathsf{aut}}_{\mathsf{sh}}$ be redundancy-free. Then by Theorem 3.3.6 we already have $f \in \mathcal{F}^{\mathsf{aut}}_{\mathsf{pas}}$.

We now follow the proof of [KMQ08]. Let

$$\begin{aligned}
\mathcal{P}_{-i} &:= \mathcal{P} \setminus \{\mathsf{P}_i\},\\
\vec{x} &:= x_1, \ldots, x_n \in \vec{\mathcal{X}} := \mathcal{X}_1 \times \cdots \times \mathcal{X}_n,\\
\vec{\mathcal{X}}_{-i} &:= \mathcal{X}_1 \times \ldots \times \mathcal{X}_{i-1} \times \mathcal{X}_{i+1} \times \cdots \times \mathcal{X}_n\\
\vec{x}_{-i} &:= x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n \in \vec{\mathcal{X}}_{-i}\\
\vec{\mathcal{X}}_{-i,j} &:= \mathcal{X}_1 \times \ldots \times \mathcal{X}_{i-1} \times \mathcal{X}_{i+1} \times \ldots \times \mathcal{X}_{j-1} \times \mathcal{X}_{j+1} \times \cdots \times \mathcal{X}_n\\
\vec{x}_{-i,j} &:= x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n \in \vec{\mathcal{X}}_{-i,j}.
\end{aligned}$$

We define an equivalence relation *renaming* on $\mathcal{F}_2$, generalizing the definition of [KMQ08] to the multi-party case. We call $f^{(1)}$ and $f^{(2)}$ are renamings of each other, denoted $f^{(1)} \equiv f^{(2)}$, iff $f^{(2)}$ is obtained from $f^{(1)}$ by locally renaming input and output values.

Definition 3.4.2 (Consistent Renaming)**.** A function $f^{(1)} \in \mathcal{F}$ is a consistent renaming of a function $f^{(2)} \in \mathcal{F}$ iff for all $i \in [n]$ there are

1. bijective maps $\sigma_i : \mathcal{X}_i^{(1)} \to \mathcal{X}_i^{(2)}$ between the respective input sets, and
2. $\forall x_i \in \mathcal{X}_i^{(1)}$ a bijective map $\sigma_{x_i} : f_i^{(1)}(x_i, \mathcal{X}_{-i}^{(1)}) \to f_i^{(2)}(\sigma_i(x_i), \mathcal{X}_{-i}^{(2)})$ between the respective output sets.

We then say $f^{(1)}$ and $f^{(2)}$ are renamings, denoted $f^{(1)} \equiv f^{(2)}$. $\Diamond$

If $f \equiv g$, then clearly functions f, g are efficiently and PFE securely mutually locally reducible, even in presence of active adversaries.

We define new functions $g, h \in \mathcal{F}$ by

$$\begin{aligned}
g_i(\vec{x}) &:= \{\vec{x}'_{-i} \in \vec{\mathcal{X}}_{-i} \mid f_i(\vec{x}) = f_i(\vec{x}'_{-i}, x_i)\},\\
h_i(\vec{x}) &:= (g_j(\vec{x}))_{j \in [n]},
\end{aligned}$$

where h is by construction symmetric. Clearly $f \equiv g$. We now show that $g \equiv h$. It suffices to prove

$$\begin{aligned}
&\forall i \in [n]\, \forall x_i \in \mathcal{X}_i,\ \vec{x}_{-i}, \vec{x}'_{-i} \in \mathcal{X}_{-i} :\\
&\quad g_i(x_i, \vec{x}_{-i}) = g_i(x_i, \vec{x}'_{-i}) \iff h_i(x_i, \vec{x}_{-i}) = h_i(x_i, \vec{x}'_{-i})
\end{aligned}$$

By definition of h, this is equivalent to

$$\forall i \in [n]\, \forall x_i \in \mathcal{X}_i,\ \vec{x}_{-i}, \vec{x}'_{-i} \in \mathcal{X}_{-i} : \\ g_i(x_i, \vec{x}_{-i}) = g_i(x_i, \vec{x}'_{-i}) \implies g_{-i}(x_i, \vec{x}_{-i}) = g_{-i}(x_i, \vec{x}'_{-i}) \tag{3.23}$$

Towards a contradiction assume we have a party $\mathsf{P}_i \in \mathcal{P}$ and inputs $x_i \in \mathcal{X}_i$ and $\vec{x}_{-i}, \vec{x}'_{-i} \in \mathcal{X}_{-i}$ such that for inputs $\vec{x} := (x_i, \vec{x}_{-i})$ and $\vec{x}' := (x_i, \vec{x}'_{-i})$ we have $g_i(\vec{x}) = g_i(\vec{x}')$ but $g_{-i}(\vec{x}) \neq g_{-i}(\vec{x}')$. So there is a party $\mathsf{P}_j \in \mathsf{P}_{-i}$ such that $g_j(\vec{x}) \neq g_j(\vec{x}')$.

Because of $g_j(\vec{x}) \neq g_j(\vec{x}')$ we find an input $\vec{x}''_{-j} \in \mathcal{X}_{-j}$ such that wlog (else interchange $\vec{x}$ and $\vec{x}'$) $\vec{x}''_{-j} \in g_j(\vec{x}')$, $\vec{x}''_{-j} \notin g_j(\vec{x})$. We now define $\vec{x}'' := (x_j, \vec{x}''_{-j})$, $\vec{x}''' := (x'_j, \vec{x}''_{-j})$, and

$$\vec{\mathcal{X}}' := \{x_i, x''_i\} \times \{x_j, x'_j\} \times \{\vec{x}_{-i,j}, \vec{x}'_{-i,j}, \vec{x}''_{-i,j}\}$$

and claim that $f\mid_{\vec{\mathcal{X}}'} \notin \mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$. By definition of g and choice of $\vec{x}$, $\vec{x}'$, $\vec{x}''$, $\vec{x}'''$ we have

$$f_i(\vec{x}) = f_i(\vec{x}') \qquad \text{by choice of } \vec{x} = (x_i, \vec{x}_{-i}),\ \vec{x}' = (x_i, \vec{x}'_{-i}) \tag{3.24}$$

$$f_j(\vec{x}) \neq f_j(\vec{x}'') \qquad \text{because } x''_{-j} \notin g_j(\vec{x}) \tag{3.25}$$

$$f_j(\vec{x}') = f_j(\vec{x}''') \qquad \text{because } x''_{-j} \in g_j(\vec{x}'). \tag{3.26}$$

So $f\mid_{\vec{\mathcal{X}}'}$ is not locally computable due to $f_j(\vec{x}) \neq f_j(\vec{x}'')$, and there is no K-cut for P_i due to $f_j(\vec{x}') = f_j(\vec{x}''')$, no K-cut for P_j due to $f_i(\vec{x}) = f_i(\vec{x}')$, and no K-cut for $\mathsf{P}_{-i,j}$ due to $f_i(\vec{x}) = f_i(\vec{x}')$ and $f_j(\vec{x}') = f_j(\vec{x}''')$.

So by Theorem3.2.6 we have $f\mid_{\vec{\mathcal{X}}'} \notin \mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$ and moreover $f\mid_{\vec{\mathcal{X}}'}$ being a restriction of f we find $f \notin \mathcal{F}_{\mathsf{pas}}^{\mathsf{aut}}$ in contradiction to the choice of f. We conclude that Equation (3.23) holds and thus $f \equiv h =: \mathrm{sym}(f)$. The statement of Lemma 3.4.1 immediately follows. □

3.5 The Class $\mathcal{F}_{\mathsf{act}}^{\mathsf{aut}}$ of Actively Computable Functions

In contrast to previous sections some of the results in this section are limited to the 2-party case. We characterize the class $\mathcal{F}_{\mathsf{2act}}$ of 2-party functions which can securely be computed in presence of an unbounded active adversary. We give some generalizations to the n-party scenario, but our 2-party results are sufficient to see that $\mathcal{F}_{\mathsf{2act}}$ is strictly contained in $\mathcal{F}_{\mathsf{2sh}}$ and hence the notion of LT security lies strictly between IT security and CO security.

Definition 3.5.1 ($\mathcal{F}_{\mathsf{act}}^{\mathsf{aut}}$: Actively Computable Functions)**.** The class of *actively computable* functions $\mathcal{F}_{\mathsf{act}}^{\mathsf{aut}}$ consists of the functions $f \in \mathcal{F}$ for which a constant round protocol $\pi \in \mathsf{Poly}$ exists that implements F_f with IT security in presence of an active adversary in the authenticated channels model with broadcast. $\Diamond$

Note that by Lemma 2.8.3 we have $\mathcal{F}_{\mathsf{act}}^{\mathsf{aut}} = \mathcal{F}_{\mathsf{act}}^{\mathsf{bc}}$, where $\mathcal{F}_{\mathsf{act}}^{\mathsf{bc}}$ denotes the functions computable by public discussion in the setting above. Hence we may, for the sake of simplicity, assume an authenticated BC channel as the sole underlying resource in the following discussion.

Interestingly there are some useful functions in the class $\mathcal{F}_{\mathsf{act}}^{\mathsf{aut}}$, e.g. $f^{(7)}$ in Fig. 3.2 which is a formalization of a Dutch flower auction, where the price is lowered in every round until a party decides to buy.

We next give a combinatorial characterization of actively computable functions, which essentially states that a party P_i must be able to send a message about its input such that the corrupted parties $\mathcal{A}$ reacting to this new information by changing their input from $x'_{\mathcal{A}}$ to $x''_{\mathcal{A}}$ could have achieved the same effect on the output by selecting a third input $x_{\mathcal{A}}$ a priori:

Definition 3.5.2 ($\mathcal{F}'_{\mathsf{act}}$: Actively Decomposable Functions)**.** A function $f \in \mathcal{F}$ is called *actively decomposable*, denoted $f \in \mathcal{F}'_{\mathsf{act}}$, if and only if $\hat{f} \in \widehat{\mathcal{F}_{\mathsf{act}}}$. We have $f \in \widehat{\mathcal{F}_{\mathsf{act}}}$ if one of the following holds:

1. f is locally computable ($f \in \mathcal{F}_{\mathsf{loc}}$);
2. there is an $i \in [n]$ and a partition (T-Cut) of $\mathcal{X}_i$ into non-empty sets $\mathcal{X}'_i \dot\cup \mathcal{X}''_i = \mathcal{X}_i$ such that
   (i) $f|_{\mathcal{X}_1 \times \ldots \times \mathcal{X}'_i \times \ldots \times \mathcal{X}_n}, f|_{\mathcal{X}_1 \times \ldots \times \mathcal{X}''_i \times \ldots \times \mathcal{X}_n} \in \widehat{\mathcal{F}_{\mathsf{act}}}$ and
   (ii) for all $\mathcal{A} \subseteq \mathcal{P} \setminus \{\mathsf{P}_i\}$ and $\mathcal{H}' := \mathcal{H} \setminus \{\mathsf{P}_i\}$ we have a K-cut

$$\forall \bar{x}_{\mathcal{A}} \in \mathcal{X}_{\mathcal{A}} : f_{\mathcal{A}}(\bar{x}_{\mathcal{A}}, \mathcal{X}_{\mathcal{H}'}, \mathcal{X}'_i) \cap f_{\mathcal{A}}(\bar{x}_{\mathcal{A}}, \mathcal{X}_{\mathcal{H}'}, \mathcal{X}''_i) = \emptyset$$

   as in Definition 3.2.4 and

$$\begin{aligned} &\forall x'_{\mathcal{A}}, x''_{\mathcal{A}} \in \mathcal{X}_{\mathcal{A}}\ \exists x_{\mathcal{A}} \in \mathcal{X}_{\mathcal{A}}\ \forall x_{\mathcal{H}'} \in \mathcal{X}_{\mathcal{H}'} \\ &(\qquad \forall x'_i \in \mathcal{X}'_i : f_{\mathcal{H}}(x'_{\mathcal{A}}, x_{\mathcal{H}'}, x'_i) = f_{\mathcal{H}}(x_{\mathcal{A}}, x_{\mathcal{H}'}, x'_i) \\ &\quad \wedge \quad \forall x''_i \in \mathcal{X}''_i : f_{\mathcal{H}}(x''_{\mathcal{A}}, x_{\mathcal{H}'}, x''_i) = f_{\mathcal{H}}(x_{\mathcal{A}}, x_{\mathcal{H}'}, x''_i)\ ). \end{aligned}$$

$\Diamond$

Active decomposability indeed characterizes the actively computable functions.

Theorem 3.5.3. *A function $f \in \mathcal{F}$ is actively computable if it is actively decomposable. In short $\mathcal{F}_{\mathsf{act}}^{\mathsf{aut}} \supseteq \mathcal{F}'_{\mathsf{act}}$. In the 2-party case[17] we even have $\mathcal{F}_{2\mathsf{act}} \subseteq \mathcal{F}'_{2\mathsf{act}}$, i.e. $\mathcal{F}_{2\mathsf{act}} = \mathcal{F}'_{2\mathsf{act}}$. Furthermore, any function $f \in \mathcal{F}'_{\mathsf{act}}$ can be computed efficiently with PFE security.*

Furthermore, we conjecture:

Conjecture 3.5.4. *A function $f \in \mathcal{F}$ is actively computable if and only if it is actively decomposable. In short $\mathcal{F}_{\mathsf{act}}^{\mathsf{aut}} = \mathcal{F}'_{\mathsf{act}}$. Furthermore, any function $f \in \mathcal{F}_{\mathsf{act}}^{\mathsf{aut}}$ can be computed efficiently with PFE security.*

The proof of Theorem 3.5.3 can be found below. The implication $f \in \mathcal{F}'_{\mathsf{act}} \implies f \in \mathcal{F}_{\mathsf{act}}^{\mathsf{aut}}$ is proven by showing the protocol for the semi-honest scenario secure against active adversaries, when applied to the T-cuts of a function $f \in \mathcal{F}'_{\mathsf{act}}$ instead of the K-cuts of a function in $\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}}$. To obtain $f \in \mathcal{F}_{2\mathsf{act}} \implies f \in \mathcal{F}'_{2\mathsf{act}}$ we observe that for $f \notin \mathcal{F}'_{2\mathsf{act}}$ the adversary can in any protocol induce an output distribution that is impossible to achieve in the ideal setting. The adversary does this by extracting information on the inputs of other participants from the protocol messages and adjusting his input according to that information.

Note that Maji et al. [MPR09] have meanwhile proven that a symmetric two party function f is computable with IT security (even using an inefficient protocol) if and only if $f \in \mathcal{F}'_{2\mathsf{act}}$. Their result is easily extended to asymmetric functions by using the symmetrization of Section 3.4.

Proof of Theorem 3.5.3. By Lemma 3.3.4 it is sufficient to show for redundancy-free functions f where $f = \hat{f}$ that $f \in \mathcal{F}'_{\mathsf{act}} \implies f \in \mathcal{F}_{\mathsf{act}}^{\mathsf{aut}}$ and $f \in \mathcal{F}_{2\mathsf{act}} \implies f \in \mathcal{F}'_{2\mathsf{act}}$. For all other (not redundancy-free) functions we then find

$$f \in \mathcal{F}'_{\mathsf{act}} \overset{\text{Def. 3.5.2}}{\iff} \hat{f} \in \mathcal{F}'_{\mathsf{act}} \implies \hat{f} \in \mathcal{F}_{\mathsf{act}}^{\mathsf{aut}} \overset{\text{Lem. 3.3.4}}{\iff} f \in \mathcal{F}_{\mathsf{act}}^{\mathsf{aut}}$$
$$f \in \mathcal{F}_{2\mathsf{act}} \overset{\text{Lem. 3.3.4}}{\iff} \hat{f} \in \mathcal{F}_{2\mathsf{act}} \implies \hat{f} \in \mathcal{F}'_{2\mathsf{act}} \overset{\text{Def. 3.5.2}}{\iff} f \in \mathcal{F}'_{2\mathsf{act}}.$$

So in the following let $f = \hat{f}$ be redundancy free.

[17] For a function class $\mathcal{F}_{\mathsf{name}}^{\mathsf{chan}}$ we denote the 2-party subclass $\mathcal{F}_{\mathsf{name}}^{\mathsf{chan}} \cap \mathcal{F}_2$ by $\mathcal{F}_{2\mathsf{name}}$. We drop the communication model specification chan as it is irrelevant for the 2-party setting.

$f \in \mathcal{F}'_{\mathsf{act}} \implies f \in \mathcal{F}^{\mathsf{aut}}_{\mathsf{act}}$**:** This part of the proof is considerably simplified by use of symmetrization as described in Lemma 3.4.1. Lemma 3.4.1 states that any function $f \in \mathcal{F}^{\mathsf{aut}}_{\mathsf{sh}}$ is mutually locally reducible to a symmetric, redundancy free function $\mathrm{sym}(f)$, i.e. for all $i, j \in [n]$ we have $\mathrm{sym}(f)_i = \mathrm{sym}(f)_j$. Now it is easy to see that $\mathcal{F}'_{\mathsf{act}}, \mathcal{F}^{\mathsf{aut}}_{\mathsf{act}} \subseteq \mathcal{F}^{\mathsf{aut}}_{\mathsf{sh}}$ and it suffices to show that $\mathrm{sym}(f) \in \mathcal{F}'_{\mathsf{act}} \implies \mathrm{sym}(f) \in \mathcal{F}^{\mathsf{aut}}_{\mathsf{act}}$ because using Lemma 3.4.1 we have

$$f \in \mathcal{F}'_{\mathsf{act}} \overset{(*)}{\iff} \mathrm{sym}(f) \in \mathcal{F}'_{\mathsf{act}} \implies \mathrm{sym}(f) \in \mathcal{F}^{\mathsf{aut}}_{\mathsf{act}} \overset{\text{Lem. 3.4.1}}{\iff} f \in \mathcal{F}^{\mathsf{aut}}_{\mathsf{act}}.$$

Implication $(*)$ holds as we have seen in the proof of Lemma 3.4.1, that $\mathrm{sym}(f)$ is a consistent renaming of $\hat{f}$ so

$$\mathrm{sym}(f) \in \mathcal{F}'_{\mathsf{act}} \iff \hat{f} \in \mathcal{F}'_{\mathsf{act}} \overset{\text{Def. 3.5.2}}{\iff} f \in \mathcal{F}'_{\mathsf{act}}.$$

So wlog let $f \in \mathcal{F}'_{\mathsf{act}}$ be symmetric and redundancy-free.

We prove $f \in \mathcal{F}'_{\mathsf{act}} \implies f \in \mathcal{F}^{\mathsf{bc}}_{\mathsf{act}} \overset{\text{Lem. 2.8.3}}{=} \mathcal{F}^{\mathsf{aut}}_{\mathsf{act}}$ by induction over the size of the input space $|\mathcal{X}_1 \times \ldots \times \mathcal{X}_n|$. Now $f \in \mathcal{F}'_{\mathsf{act}}$ implies that f is either locally computable ($f \in \mathcal{F}_{\mathsf{loc}}$), in which case f is clearly actively computable ($f \in \mathcal{F}^{\mathsf{bc}}_{\mathsf{act}}$) and we are done, or f has a T-cut as defined in Definition 3.5.2.

In the second case, by definition of the class $\mathcal{F}'_{\mathsf{act}}$ for some $\mathsf{P}_i \in \mathcal{P}$ the input set $\mathcal{X}_i$ has a partition $\mathcal{X}_i^{(1)} \dot\cup \mathcal{X}_i^{(2)} = \mathcal{X}_i$ such that

(i) $f^{(1)} := f|_{\mathcal{X}_1 \times \ldots \times \mathcal{X}_i^{(1)} \times \ldots \times \mathcal{X}_n}, f^{(2)} := f|_{\mathcal{X}_1 \times \ldots \times \mathcal{X}_i^{(2)} \times \ldots \times \mathcal{X}_n} \in \mathcal{F}'_{\mathsf{act}}$ and

(ii) $\forall \mathcal{A} \subseteq \mathcal{P} \setminus \{\mathsf{P}_i\}$ and $\mathcal{H}' = \mathcal{H} \setminus \{\mathsf{P}_i\}$ we have

$$\forall \bar{x}_{\mathcal{A}} \in \mathcal{X}_{\mathcal{A}} : f_{\mathcal{A}}(\bar{x}_{\mathcal{A}}, \mathcal{X}_{\mathcal{H}'}, \mathcal{X}_i^{(1)}) \cap f_{\mathcal{A}}(\bar{x}_{\mathcal{A}}, \mathcal{X}_{\mathcal{H}'}, \mathcal{X}_i^{(2)}) = \emptyset$$

and $\forall x'_{\mathcal{A}}, x''_{\mathcal{A}} \in \mathcal{X}_{\mathcal{A}}\ \exists x_{\mathcal{A}} \in \mathcal{X}_{\mathcal{A}}\ \forall x_{\mathcal{H}'} \in \mathcal{X}_{\mathcal{H}'}$

$$\begin{aligned} \forall x_i^{(1)} \in \mathcal{X}_i^{(1)} : \quad & f_{\mathcal{H}}(x'_{\mathcal{A}}, x_{\mathcal{H}'}, x_i^{(1)}) = f_{\mathcal{H}}(x_{\mathcal{A}}, x_{\mathcal{H}'}, x_i^{(1)}) \qquad \wedge \\ \forall x_i^{(2)} \in \mathcal{X}_i^{(2)} : \quad & f_{\mathcal{H}}(x''_{\mathcal{A}}, x_{\mathcal{H}'}, x_i^{(2)}) = f_{\mathcal{H}}(x_{\mathcal{A}}, x_{\mathcal{H}'}, x_i^{(2)}) \end{aligned}$$

By induction hypothesis we then have $f^{(1)}, f^{(2)} \in \mathcal{F}^{\mathsf{bc}}_{\mathsf{act}}$ as

$$|\mathcal{X}_1 \times \ldots \times \mathcal{X}_i^{(1)} \times \ldots \times \mathcal{X}_n| < |\mathcal{X}_1 \times \ldots \times \mathcal{X}_i \times \ldots \times \mathcal{X}_n|$$

$$|\mathcal{X}_1 \times \ldots \times \mathcal{X}_i^{(2)} \times \ldots \times \mathcal{X}_n| < |\mathcal{X}_1 \times \ldots \times \mathcal{X}_i \times \ldots \times \mathcal{X}_n|$$

and there are protocols $\pi^{(1)}, \pi^{(2)}$ actively PFE securely implementing $\mathsf{F}_{f^{(1)}}, \mathsf{F}_{f^{(2)}}$ under simulators $\mathsf{S}^{(1)}, \mathsf{S}^{(2)}$.

We now construct a protocol π by adding a first protocol round where party P_i broadcasts a message $m_1 \in \{1, 2\}$ indicating whether P_i's input x_i is $x_i \in \mathcal{X}_i^{(1)}$ or $x_i \in \mathcal{X}_i^{(2)}$. If party P_i sends no message, the other parties just assume $m_1 = 1$. Subsequently the parties proceed to run $\pi^{(m_1)}$.

It remains to prove that this protocol π is PFE secure by providing an appropriate simulator $\mathsf{S} \in \mathfrak{S}_{\mathsf{act}}$ for the case where an arbitrary subset $\mathcal{A}$ of the parties $\mathcal{P}$ is corrupted by the adversary $\mathsf{A} \in \mathfrak{A}_{\mathsf{act}}$. It suffices to consider two cases. Either P_i is in the set of corrupted parties ($\mathsf{P}_i \in \mathcal{A}$), or it is not ($\mathsf{P}_i \notin \mathcal{A}$).

Let us first consider the case where $\mathsf{P}_i \notin \mathcal{A}$. We construct the simulator $\mathsf{S}_{\mathsf{P}_i \notin \mathcal{A}}$ as follows:

1. $\mathsf{S}_{\mathsf{P}_i \notin \mathcal{A}}$ fixes the randomness of the adversary A
2. It feeds the input x_{A} of the distinguisher D to A
3. It generates two copies $\mathsf{A}^{(1)}$ and $\mathsf{A}^{(2)}$ of A and feeds $m_1 = 1$ to $\mathsf{A}^{(1)}$ and $m_1 = 2$ to $\mathsf{A}^{(2)}$
4. Now $\mathsf{S}_{\mathsf{P}_i \notin \mathcal{A}}$ runs $\mathsf{S}^{(1)}(\mathsf{A}^{(1)})$ and $\mathsf{S}^{(2)}(\mathsf{A}^{(2)})$ until these simulators produce inputs $x_{\mathcal{A}}^{(1)}$ and $x_{\mathcal{A}}^{(2)}$ for $\mathsf{F}_{f^{(1)}}$ and $\mathsf{F}_{f^{(2)}}$ respectively
5. Since $f \in \mathcal{F}'_{\mathsf{act}}$ we know:

$$\forall \bar{x}_{\mathcal{A}} \in \mathcal{X}_{\mathcal{A}} : f_{\mathcal{A}}(\bar{x}_{\mathcal{A}}, \mathcal{X}_{\mathcal{H}'}, \mathcal{X}_i^{(1)}) \cap f_{\mathcal{A}}(\bar{x}_{\mathcal{A}}, \mathcal{X}_{\mathcal{H}'}, \mathcal{X}_i^{(2)}) = \emptyset \tag{3.27}$$

and $\exists x_{\mathcal{A}} \in \mathcal{X}_{\mathcal{A}}\ \forall x_{\mathcal{H}'} \in \mathcal{X}_{\mathcal{H}'}$

$$\begin{aligned} &\forall x_i^{(1)} \in \mathcal{X}_i^{(1)} : \quad f_{\mathcal{H}}(x_{\mathcal{A}}^{(1)}, x_{\mathcal{H}'}, x_i^{(1)}) = f_{\mathcal{H}}(x_{\mathcal{A}}, x_{\mathcal{H}'}, x_i^{(1)}) \quad \wedge \\ &\forall x_i^{(2)} \in \mathcal{X}_i^{(2)} : \quad f_{\mathcal{H}}(x_{\mathcal{A}}^{(2)}, x_{\mathcal{H}'}, x_i^{(2)}) = f_{\mathcal{H}}(x_{\mathcal{A}}, x_{\mathcal{H}'}, x_i^{(2)}) \end{aligned} \tag{3.28}$$

The simulator $\mathsf{S}_{\mathsf{P}_i \notin \mathcal{A}}$ computes this $x_{\mathcal{A}}$, inputs it to F_f and receives one of the following two outputs:

$$\begin{aligned} y_{\mathcal{A}} &= f_{\mathcal{A}}(x_{\mathcal{A}}, x_{\mathcal{H}'}, x_i) \overset{(*)}{=} f_{\mathcal{H}}(x_{\mathcal{A}}, x_{\mathcal{H}'}, x_i) \\ &\overset{(3.28)}{=} f_{\mathcal{H}}(x_{\mathcal{A}}^{(1)}, x_{\mathcal{H}'}, x_i) \overset{(*)}{=} f_{\mathcal{A}}(x_{\mathcal{A}}^{(1)}, x_{\mathcal{H}'}, x_i) \text{ if } x_i \in \mathcal{X}_i^{(1)} \\ y_{\mathcal{A}} &= f_{\mathcal{A}}(x_{\mathcal{A}}, x_{\mathcal{H}'}, x_i) \overset{(*)}{=} f_{\mathcal{H}}(x_{\mathcal{A}}, x_{\mathcal{H}'}, x_i) \\ &\overset{(3.28)}{=} f_{\mathcal{H}}(x_{\mathcal{A}}^{(2)}, x_{\mathcal{H}'}, x_i) \overset{(*)}{=} f_{\mathcal{A}}(x_{\mathcal{A}}^{(2)}, x_{\mathcal{H}'}, x_i) \text{ if } x_i \in \mathcal{X}_i^{(2)} \end{aligned} \tag{3.29}$$

where the equalities marked by $(*)$ follow from the symmetry of the function f, which we assumed in the beginning of the proof.

6. Now if $y_{\mathcal{A}} \in f_{\mathcal{A}}(x_{\mathcal{A}}, \mathcal{X}_{\mathcal{H}'}, \mathcal{X}_i^{(1)})$ (by Equation (3.27) this check is always unambiguous) then simulator $\mathsf{S}_{\mathsf{P}_i \notin \mathcal{A}}$ passes output $y_{\mathcal{A}}$ to subsimulator $\mathsf{S}^{(1)}(\mathsf{A}^{(1)})$ and returns the output of that simulator. Note that the above Equation (3.29) shows that the input $y_{\mathcal{A}}$ is correct for $\mathsf{S}^{(1)}(\mathsf{A}^{(1)})$.
   If $y_{\mathcal{A}} \in f_{\mathcal{A}}(x_{\mathcal{A}}, \mathcal{X}_{\mathcal{H}'}, \mathcal{X}_i^{(2)})$ then simulator $\mathsf{S}_{\mathsf{P}_i \notin \mathcal{A}}$ proceeds in the same way with subsimulator $\mathsf{S}^{(2)}(\mathsf{A}^{(2)})$.

The correctness of the above simulation is obvious.

We now turn to the second case, where party P_i is corrupted, i.e. we have $\mathsf{P}_i \in \mathcal{A}$. The simulator $\mathsf{S}_{\mathsf{P}_i \in \mathcal{A}}$ can be constructed as follows:

1. Simulator $\mathsf{S}_{\mathsf{P}_i \in \mathcal{A}}$ fixes the randomness of the adversary A.
2. It passes the input x_{A} provided by the distinguisher D to A.
3. It runs the adversary A until A outputs a message m_1.
4. If the adversary makes no output $m_1 \in \{1, 2\}$, simulator $\mathsf{S}_{\mathsf{P}_i \in \mathcal{A}}$ uses the message $m_1 = 1$.
5. Now the simulator $\mathsf{S}_{\mathsf{P}_i \in \mathcal{A}}$ runs the subsimulator $\mathsf{S}^{(m_1)}(\mathsf{A})$ and simply forwards all inputs and outputs.

This simulates the real execution, as depending on the message m_1 the protocol $\pi^{(m_1)}$ is executed by the honest parties. The interaction with this protocol is then faithfully simulated by subsimulator $\mathsf{S}^{(m_1)}(\mathsf{A})$.

Note that we actually prove active PFE security above. The computation of locally computable functions is perfectly secure (induction base) and in the above argument if the subsimulators are perfectly secure and efficient, so are the resulting simulators (induction step).

$f \in \mathcal{F}_{\mathsf{2act}} \implies f \in \mathcal{F}'_{\mathsf{2act}}$: As we are considering the two party setting now we will refer to the parties as Alice, denoted A, and Bob, denoted B.

Recall that wlog function f is redundancy-free ($f = \hat{f}$). If function f is locally computable ($f \in \mathcal{F}_{\mathsf{2loc}}$) then $f \in \mathcal{F}'_{\mathsf{2act}}$ and we are done. So consider an $f \notin \mathcal{F}_{\mathsf{2loc}}$. We show that f has a T-cut as defined in Definition 3.5.2 and then apply an inductive argument. As function f is actively

computable we have $f \in \mathcal{F}_{\mathsf{2sh}}$ and as $f \notin \mathcal{F}_{\mathsf{2loc}}$ function f has a K-cut. Wlog we have a partition of Bob's input set $\mathcal{X}_B = \mathcal{X}'_B \cup \mathcal{X}''_B$ and

$$\forall x_A \in \mathcal{X}_A \ : \ f_A(x_A, \mathcal{X}'_B) \cap f_A(x_A, \mathcal{X}''_B) = \emptyset$$

Now for any $x_A \in \mathcal{X}_A$ and regardless of the a priori distribution $\Pr_{X_A,X_B}$ on the inputs, as the function outputs on $\mathcal{X}'_B$ and $\mathcal{X}''_B$ differ, so must the protocol outputs y_A for A with overwhelming probability. As a result the statistical distance of the transcripts under input taken from $x_A, \mathcal{X}'_B$ or $x_A, \mathcal{X}''_B$ must be overwhelming as well:

$$\begin{aligned} 1 - \mathsf{negl} &< \Delta(\Pr_{Y_A|X_A,X_B}(\cdot, x_A, \mathcal{X}'_B), \Pr_{Y_A|X_A,X_B}(\cdot, x_A, \mathcal{X}''_B)) \\ &\leq \Delta(\Pr_{T|X_A,X_B}(\cdot, x_A, \mathcal{X}'_B), \Pr_{T|X_A,X_B}(\cdot, x_A, \mathcal{X}''_B)). \end{aligned}$$

Note, that being independent of the a priori input distribution $\Pr_{X_A,X_B}$, this result applies to all subsets of the input sets $\mathcal{X}'_B$, $\mathcal{X}''_B$ as well. In particular for any inputs $x'_B \in \mathcal{X}'_B$, $x''_B \in \mathcal{X}''_B$, $x_A \in \mathcal{X}_A$ we have

$$\begin{aligned} 1 - \mathsf{negl} &< \Delta(\Pr_{Y_A|X_A,X_B}(\cdot, x_A, x'_B), \Pr_{Y_A|X_A,X_B}(\cdot, x_A, x''_B)) \\ &\leq \Delta(\Pr_{T|X_A,X_B}(\cdot, x_A, x'_B), \Pr_{T|X_A,X_B}(\cdot, x_A, x''_B)). \end{aligned}$$

Now for any $\mathcal{X}'_B \cup \mathcal{X}''_B = \mathcal{X}_B$, $a \in \mathcal{X}_A$

$$\Delta(\Pr_{T_0|X_A,X_B}(\cdot, a, \mathcal{X}'_B), \Pr_{T_0|X_A,X_B}(\cdot, a, \mathcal{X}''_B)) = 0.$$

Hence considering all possible choices of values $x_A \in \mathcal{X}_A$ and cuts $\mathcal{X}'_B \dot\cup \mathcal{X}''_B = \mathcal{X}_B$, as well as symmetrically $x_B \in \mathcal{X}_B$ and cuts $\mathcal{X}'_A \cup \mathcal{X}''_A = \mathcal{X}_A$, there must be a minimal round number r for which there is a distribution $\Pr_{X_A,X_B}$ on the inputs such that we have

$$\begin{aligned} \alpha :=& \Delta(\Pr_{T_r|X_A,X_B}(\cdot, x_A, \mathcal{X}'_B), \Pr_{T_r|X_A,X_B}(\cdot, x_A, \mathcal{X}''_B)) > \mathsf{ntcbl} \qquad \text{or} \\ & \Delta(\Pr_{T_r|X_A,X_B}(\cdot, \mathcal{X}'_A, x_B), \Pr_{T_r|X_A,X_B}(\cdot, \mathcal{X}''_A, x_B)) > \mathsf{ntcbl} \end{aligned} \tag{3.30}$$

Wlog we assume that the minimum r is achieved for some particular $x_A \in \mathcal{X}_A$ and a cut $\mathcal{X}'_B \cup \mathcal{X}''_B = \mathcal{X}_B$. Then, by minimality of the round number r, for all $r' < r$

$$\begin{aligned} &\forall x_B \in \mathcal{X}_B,\ x_A \in \mathcal{X}_A,\ \mathcal{X}'_B \dot\cup \mathcal{X}''_B = \mathcal{X}_B\ \mathcal{X}'_A \dot\cup \mathcal{X}''_A = \mathcal{X}_A \ : \\ &\quad \Delta(\Pr_{T_{r'}|X_A,X_B}(\cdot, x_A, \mathcal{X}'_B), \Pr_{T_{r'}|X_A,X_B}(\cdot, x_A, \mathcal{X}''_B)) < \mathsf{negl} \qquad \text{and} \\ &\quad \Delta(\Pr_{T_r|X_A,X_B}(\cdot, \mathcal{X}'_A, x_B), \Pr_{T_r|X_A,X_B}(\cdot, \mathcal{X}''_A, x_B)) < \mathsf{negl}. \end{aligned} \tag{3.31}$$

So the message m_r travels from B to A, as A cannot construct m_r for lack of information about x_B. So Equation (3.31) and Equation (3.30) hold

for the given cut $\mathcal{X}'_B \cup \mathcal{X}''_B = \mathcal{X}_B$ and an *arbitrary* $x_A \in \mathcal{X}_A$, as m_r is constructed by B without information about x_A.

Furthermore note that we can find a cut $\mathcal{X}'_B \dot{\cup} \mathcal{X}''_B = \mathcal{X}_B$ such that the above holds for every distribution $\Pr_{X_A,X_B}$ on the inputs. This can be seen using a hybrid argument over the inputs.

We need to show now

1. $\mathcal{X}'_B \dot{\cup} \mathcal{X}''_B = \mathcal{X}_B$ constitutes a K-cut
2. $\mathcal{X}'_B \dot{\cup} \mathcal{X}''_B = \mathcal{X}_B$ constitutes a T-cut
3. the argument perpetuates inductively

The cut $\mathcal{X}'_B \dot{\cup} \mathcal{X}''_B = \mathcal{X}_B$ constitutes a K-cut. Towards a contradiction assume we have $b' \in \mathcal{X}'_B$, $b'' \in \mathcal{X}''_B$, $a \in \mathcal{X}_A$ such that $f_A(a, b') = f_A(a, b'')$. Then for any simulated transcript T_{S}

$$\Pr_{T_{\mathsf{S}}|X_A,X_B}(\cdot, a, b') = \Pr_{T_{\mathsf{S}}|X_A,X_B}(\cdot, a, b'')$$

Now by security there exists a simulator S such that

$$\Delta(\Pr_{T_{\mathsf{S}}|X_A,X_B}(\cdot, a, b'), \Pr_{T|X_A,X_B}(\cdot, a, b')) < \mathsf{negl},$$
$$\Delta(\Pr_{T_{\mathsf{S}}|X_A,X_B}(\cdot, a, b''), \Pr_{T|X_A,X_B}(\cdot, a, b'')) < \mathsf{negl}.$$

Then by transitivity

$$\Delta(\Pr_{T|X_A,X_B}(\cdot, a, b'), \Pr_{T|X_A,X_B}(\cdot, a, b'')) < \mathsf{negl}$$

which implies

$$\Delta(\Pr_{T_r|X_A,X_B}(\cdot, a, b'), \Pr_{T_r|X_A,X_B}(\cdot, a, b'')) < \mathsf{negl}$$

in contradiction to Equation (3.30).

The cut $\mathcal{X}'_B \dot{\cup} \mathcal{X}''_B = \mathcal{X}_B$ is a T-cut. Consider an $a'' \in \mathcal{X}_A$ and the cut $\mathcal{X}'_B \dot{\cup} \mathcal{X}''_B = \mathcal{X}_B$ from Equation (3.30). From Equation (3.31) we have for any $a' \in \mathcal{X}_A$

$$\Delta(\Pr_{T_r|X_A,X_B}(\cdot, a', \mathcal{X}'_B), \Pr_{T_r|X_A,X_B}(\cdot, a'', \mathcal{X}'_B)) < \mathsf{negl}.$$

Thus at round r the distribution of transcripts for different inputs of party A differs still only negligibly. So with overwhelming probability

a corrupted A can choose (uniformly among the matching ones) a new random string c'_A such that under the random string c'_A the input a' is consistent with the transcript T_r observed so far. The execution then proceeds according to protocol π with the new values a' as input and c'_A as random string. We now fix an $a' \in \mathcal{X}_A$ and name the procedure just described π' and the induced transcript random variable T'.

We find that the distributions on the transcript T' (and thus also the output distributions) resulting from this procedure differ only negligibly from the transcript distributions obtained by initiating a normal protocol run with input a':

$$
\begin{aligned}
&\Delta(\mathrm{Pr}_{T|X_A,X_B}(\cdot,a',\mathcal{X}'_B),\mathrm{Pr}_{T'|X_A,X_B}(\cdot,a'',\mathcal{X}'_B))\\
&=\sum_{t\in\Pi}|\mathrm{Pr}_{T|X_A,X_B}(t,a',\mathcal{X}'_B)-\mathrm{Pr}_{T'|X_A,X_B}(t,a'',\mathcal{X}'_B)|\\
&=\sum_{t\in\Pi}|\mathrm{Pr}_{T|T_r,X_A,X_B}(t,t_r,a',\mathcal{X}'_B)\mathrm{Pr}_{T_r|X_A,X_B}(t,a',\mathcal{X}'_B)\\
&\qquad\qquad-\mathrm{Pr}_{T'|T'_r,X_A,X_B}(t,t_r,a'',\mathcal{X}'_B)\mathrm{Pr}_{T'_r|X_A,X_B}(t,a'',\mathcal{X}'_B)|\\
&=\sum_{t\in\Pi}|\mathrm{Pr}_{T|T_r,X_A,X_B}(t,t_r,a',\mathcal{X}'_B)\mathrm{Pr}_{T_r|X_A,X_B}(t,a',\mathcal{X}'_B)\\
&\qquad\qquad-\mathrm{Pr}_{T|T_r,X_A,X_B}(t,t_r,a',\mathcal{X}'_B)\mathrm{Pr}_{T_r|X_A,X_B}(t,a'',\mathcal{X}'_B)|\\
&=\sum_{t\in\Pi}\mathrm{Pr}_{T|T_r,X_A,X_B}(t,t_r,a',\mathcal{X}'_B)\\
&\qquad\qquad|\mathrm{Pr}_{T_r|X_A,X_B}(t,a',\mathcal{X}'_B)-\mathrm{Pr}_{T_r|X_A,X_B}(t,a'',\mathcal{X}'_B)|\\
&=\Delta(\mathrm{Pr}_{T_r|X_A,X_B}(\cdot,a',\mathcal{X}'_B),\mathrm{Pr}_{T_r|X_A,X_B}(\cdot,a'',\mathcal{X}'_B))<\mathsf{negl}.
\end{aligned}
$$

We can proceed analogously for $\mathcal{X}''_B$ and thus have

$$
\begin{aligned}
\Delta(\mathrm{Pr}_{T|X_A,X_B}(\cdot,a',\mathcal{X}'_B),\mathrm{Pr}_{T'|X_A,X_B}(\cdot,a'',\mathcal{X}'_B))<\mathsf{negl},\\
\Delta(\mathrm{Pr}_{T|X_A,X_B}(\cdot,a',\mathcal{X}''_B),\mathrm{Pr}_{T'|X_A,X_B}(\cdot,a'',\mathcal{X}''_B))<\mathsf{negl}.
\end{aligned}
$$

A corrupted party A may then mount the following attack: Party A executes protocol π with input a up to round r, obtaining a transcript t_r. Now if $\mathrm{Pr}_{T_r|X_A,X_B}(t_r,a'',\mathcal{X}'_B) > \mathrm{Pr}_{T_r|X_A,X_B}(t_r,a'',\mathcal{X}''_B)$ (assuming uniform distribution on each set $\mathcal{X}'_B$ or $\mathcal{X}''_B$ of the partition) then A runs π' and π else. The resulting output distributions are indistinguishable (negligible statistical distance) from $\pi(a',X_B)$ and $\pi(a'',X_B)$ respectively as seen above. We designate this protocol execution by $\breve{\pi}$ and the induced random variable for the transcripts $\breve{T}$. Now take a distinguisher D that

with probability $\frac{1}{2}$ each chooses X_B from $\mathcal{X}'_B$ or $\mathcal{X}''_B$ respectively with uniform distribution on each set $\mathcal{X}'_B$ or $\mathcal{X}''_B$ of the partition and provides the corrupted party A with no information.

Consider the conditional distribution of party B's output under the given adversarial party A and distinguisher D where we first assume an input $b' \in \mathcal{X}'_B$:

$$\begin{aligned}
\Pr_{Y_B|X_B}(y,b') &= \sum_{t\in\Pi} \Pr_{Y_B|\breve{T},X_B}(y,t,b')\Pr_{\breve{T}|X_B}(t,b') \\
&= \sum_{t\in\Pi} \Pr_{Y_B|T,X_B}(y,t,b')\Pr_{\breve{T}|X_B}(t,b') \\
&\overset{(3.31)}{\approx} \sum_{t\in\Pi} \Pr_{Y_B|T,X_B}(y,t,b') \\
&\qquad \left(\frac{1+\alpha}{2}\Pr_{T|X_A,X_B}(t,a',b') + \frac{1-\alpha}{2}\Pr_{T|X_A,X_B}(t,a'',b')\right) \\
&\approx \frac{1+\alpha}{2}\Pr_{Y_B|X_A,X_B}(y,a',b') + \frac{1-\alpha}{2}\Pr_{Y_B|X_A,X_B}(y,a'',b') \\
&\approx \frac{1+\alpha}{2}1_{y=f_B(a',b')} + \frac{1-\alpha}{2}1_{y=f_B(a'',b')};
\end{aligned}$$

Furthermore, for an input $b'' \in \mathcal{X}''_B$:

$$\begin{aligned}
\Pr_{Y_B|X_B}(y,b'') &= \sum_{t\in\Pi} \Pr_{Y_B|\breve{T},X_B}(y,t,b'')\Pr_{\breve{T}|X_B}(t,b'') \\
&= \sum_{t\in\Pi} \Pr_{Y_B|T,X_B}(y,t,b'')\Pr_{\breve{T}|X_B}(t,b'') \\
&\overset{(3.31)}{\approx} \sum_{t\in\Pi} \Pr_{Y_B|T,X_B}(y,t,b'') \\
&\qquad \left(\frac{1-\alpha}{2}\Pr_{T|X_A,X_B}(t,a',b'') + \frac{1+\alpha}{2}\Pr_{T|X_A,X_B}(t,a'',b'')\right) \\
&\approx \frac{1-\alpha}{2}\Pr_{Y_B|X_A,X_B}(y,a',b'') + \frac{1+\alpha}{2}\Pr_{Y_B|X_A,X_B}(y,a'',b'') \\
&\approx \frac{1-\alpha}{2}1_{y=f_B(a',b'')} + \frac{1+\alpha}{2}1_{y=f_B(a'',b'')}.
\end{aligned}$$

In the ideal setting and for the distinguisher D as described above, the input a of A as provided to the ideal functionality by the simulator is

independent of the input b of B. Thus we find for the conditional distribution of the output Y_B^I of B in the ideal case, both for $b \in \mathcal{X}_B'$ and $b \in \mathcal{X}_B''$

$$\Pr_{Y_B^I|X_B}(y,b) = \sum_{a\in\mathcal{X}_A \,:\, y=f_B(a,b)} \Pr_{X_A}(a)$$

where the simulator can only adjust $\Pr_{X_A}(a)$.

By security of the protocol (simulatability) we must have a *single* distribution $\Pr_{X_A}$ such that

$$\begin{aligned}\mathsf{negl} &> \Delta(\Pr_{Y_B|X_B}(y,b'), \Pr_{Y_B^I|X_B}(y,b'))\\ \mathsf{negl} &> \Delta(\Pr_{Y_B|X_B}(y,b''), \Pr_{Y_B^I|X_B}(y,b''))\end{aligned}$$

As we show below, we can conclude that there is an input $a \in \mathcal{X}_A$ such that $\Pr_{X_A}(a) > \mathsf{ntcbl}$ and $f_B(a,b') = f_B(a',b')$ and $f_B(a,b'') = f_B(a'',b'')$ for all inputs $b' \in \mathcal{X}_B'$ and $b'' \in \mathcal{X}_B''$:

$$\begin{aligned}\mathsf{negl} &> \Delta(\Pr_{Y_B|X_B}(y,b'), \Pr_{Y_B^I|X_B}(y,b'))\\ &= \sum_{y\in\mathcal{Y}_B} |\Pr_{Y_B|X_B}(y,b') - \Pr_{Y_B^I|X_B}(y,b')|\\ &\approx \sum_{y\in\mathcal{Y}_B} \Bigg| \frac{1+\alpha}{2} 1_{y=f(a',b')} + \frac{1-\alpha}{2} 1_{y=f_B(a'',b')}\\ &\qquad\qquad - \sum_{a\in\mathcal{X}_A \,:\, y=f_B(a,b')} \Pr_{X_A}(a) \Bigg|\\ &= \sum_{y\in\mathcal{Y}_B} \Bigg| \frac{1+\alpha}{2} 1_{y=f(a',b')} + \frac{1-\alpha}{2} 1_{y=f_B(a'',b')}\\ &\qquad\qquad - \sum_{a\in\mathcal{X}_A} \Pr_{X_A}(a) 1_{y=f_B(a,b')} \Bigg| .\end{aligned}$$

If $f_B|_{\mathcal{X}_A\times\mathcal{X}_B'}$ or $f_B|_{\mathcal{X}_A\times\mathcal{X}_B''}$ are constant, then f already has a T-cut. So we consider inputs $a', a'' \in \mathcal{X}_A$ such that there is an input $b' \in \mathcal{X}_B'$ and a input $b'' \in \mathcal{X}_B''$ where $f_B(a',b') \neq f_B(a'',b')$ and $f_B(a',b'') \neq f_B(a'',b'')$. But then the obvious solution $\Pr_{X_A}(a') \approx \frac{1+\alpha}{2}$, $\Pr_{X_A}(a'') \approx \frac{1-\alpha}{2}$, $\Pr_{X_A}(a) \approx 0$

for $a' \neq a \neq a''$ leads to a contradiction with

$$\begin{aligned}\mathsf{negl} &> \Delta(\Pr_{Y_B|X_B}(y,b''), \Pr_{Y_B^I|X_B}(y,b'')) \\ &\approx \sum_{y\in\mathcal{Y}_B} \left| \frac{1-\alpha}{2} 1_{y=f(a',b'')} + \frac{1+\alpha}{2} 1_{y=f_B(a'',b'')} \right. \\ &\qquad\qquad \left. - \sum_{a\in\mathcal{X}_A} \Pr_{X_A}(a) 1_{y=f_B(a,b'')} \right| .\end{aligned}$$

So there must be an $a \in \mathcal{X}_A$ where $\Pr_{X_A}(a) > \mathsf{ntcbl}$. But then also $f(a,b') = f(a',b')$, $f(a,b'') = f(a'',b'')$, which proves that the cut $\mathcal{X}'_B \dot\cup \mathcal{X}''_B = \mathcal{X}_B$ is a T-cut. In fact, if there is only one such a, we find

$$\begin{aligned}\Pr_{X_A}(a) &\approx \alpha, \\ \Pr_{X_A}(a') &\approx \frac{1-\alpha}{2}, \\ \Pr_{X_A}(a'') &\approx \frac{1-\alpha}{2}.\end{aligned}$$

Extending the Argument Finally we need to show that the argument presented above applies inductively until we are left with locally computable restrictions of the original function.

To this end we consider $f' = f|_{\mathcal{X}_A \times \mathcal{X}'_B}$ and $f'' = f|_{\mathcal{X}_A \times \mathcal{X}''_B}$. Unfortunately we cannot readily conclude that $f \in \mathcal{F}_{\mathsf{2act}}$ implies $f', f'' \in \mathcal{F}_{\mathsf{2act}}$. Take for instance f' and some corrupted B. Then by active computability ($f \in \mathcal{F}_{\mathsf{2act}}$) of function f there is a simulator S which indistinguishly simulates the actively secure protocol π for function f when connected to functionality F_f. On functionality $\mathsf{F}_{f'}$ however the simulation may fail, as simulator S may rely on making an input $b \in \mathcal{X}''_B$ to functionality F_f, which will be rejected by functionality $\mathsf{F}_{f'}$.

However, for our proof above we employ only a specific class $\mathfrak{A}$ of adversaries, namely semi-honest adversaries and adversaries that generally adhere to the protocol π, but possibly attempt to "switch" their input. We show that security against this class $\mathfrak{A}$ of adversaries already implies $f \in \mathcal{F}'_{\mathsf{2act}}$, so in particular $f \in \mathcal{F}_{\mathsf{2act}} \implies f \in \mathcal{F}'_{\mathsf{2act}}$. So it is sufficient to argue that both f' and f'' are indeed in the class $\mathcal{F}''_{\mathsf{2act}}$ of functions secure against this specific subclass $\mathfrak{A}$ of adversaries. Then we can proceed inductively or simply argue by contraposition: Take a function $f \in \mathcal{F}''_{\mathsf{2act}}$ where $f \notin \mathcal{F}'_{\mathsf{2act}}$ and where the input space $|\mathcal{X}_A \times \mathcal{X}_B|$ is minimal. We have shown that such a function f must have a T-cut decomposing f

into functions f' and f''. Now $f', f'' \in \mathcal{F}''_{2\mathsf{act}}$ (this we show below) and hence by minimality of f we have $f', f'' \in \mathcal{F}'_{2\mathsf{act}}$. But then $f' \in \mathcal{F}'_{2\mathsf{act}}$ by definition of $\mathcal{F}'_{2\mathsf{act}}$ in contradiction to the choice of f. We hence find

$$f \in \mathcal{F}_{2\mathsf{act}} \implies f \in \mathcal{F}''_{2\mathsf{act}} \implies f \in \mathcal{F}'_{2\mathsf{act}}.$$

So we need to show that the functions f' and f'' are indeed in the class $\mathcal{F}''_{2\mathsf{act}}$ of functions secure against the subclass of adversaries $\mathfrak{A}$ used in the proof above. Now clearly for a corrupted party A we have f' and f'' as secure as f, since the same simulators can be applied. For a corrupted party B consider the argument above for the restriction $f\restriction_{\mathcal{X}_A \times \mathcal{X}'_B}$ of function f, after party B has sent the first message m_1. In other words we exchange parties A and B in the above argument and replace set $\mathcal{X}_A$ with $\mathcal{X}'_B$ and the set $\mathcal{X}_B$ with $\mathcal{X}_A$. Then the outcome Y of the real protocol execution will be $Y \in f(\mathcal{X}_A, \mathcal{X}'_B)$ with overwhelming probability. Now $\mathcal{X}'_B \cup \mathcal{X}''_B = \mathcal{X}_B$ is a K-cut, meaning

$$\forall x_A \in \mathcal{X}_A \ : \ f(x_A, \mathcal{X}'_B) \cap f(x_A, \mathcal{X}''_B) = \emptyset.$$

But then, as simulator $\mathsf{S}(B)$ is a valid simulator for F_f under protocol π, simulator $\mathsf{S}(B)$ must with overwhelming probability give an input $b \in \mathcal{X}'_B$ to F_f. Hence simulator $\mathsf{S}(B)$ is actually a valid simulator for functionality $\mathsf{F}_{f'}$ with respect to protocol π. In other words $f' \in \mathcal{F}''_{2\mathsf{act}}$. □

The functions $f^{(7)}$ and $f^{(8)}$ in Fig. 3.2 are examples of actively computable functions. Especially compare $f^{(8)}$ with $f^{(2)} \in \mathcal{F}_{2\mathsf{sh}}$ which is *not* actively computable. The lines in the tables for $f^{(7)}$ and $f^{(8)}$ represent messages which are to be sent in the secure protocol.

3.6 Long-Term Security

Subsequently we characterize the n-party functions that can be computed LT securely (without fairness) in presence of active adversaries. LT security means we are willing to make CO assumptions, but only for the duration of the protocol interaction. Once the protocol has terminated we demand IT security. We look at different classes of LT securely computable functions, defined by different channel models. The most practical model, corresponding to the class $\mathcal{F}_{\mathsf{lts}}^{\mathsf{ins,pki}}$, is an internet-like setting, where insecure channels and a PKI are available to the parties. Furthermore we also discuss the classes $\mathcal{F}_{\mathsf{lts}}^{\mathsf{aut}}$ where authenticated channels and

$\mathcal{F}_{\mathsf{lts}}^{\mathsf{bc}}$ where an authenticated BC channel are given. We find that all these classes $\mathcal{F}_{\mathsf{lts}}^{\mathsf{ins,pki}} = \mathcal{F}_{\mathsf{lts}}^{\mathsf{bc}} = \mathcal{F}_{\mathsf{lts}}^{\mathsf{aut}}$ are equal to the class $\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}}$ of semi-honestly computable functions.

Definition 3.6.1 ($\mathcal{F}_{\mathsf{lts}}^{\mathsf{bc}}$, $\mathcal{F}_{\mathsf{lts}}^{\mathsf{ins,pki}}$, $\mathcal{F}_{\mathsf{lts}}^{\mathsf{aut}}$: LT Computable Functions)**.** The classes of *LT computable* functions

i. $\mathcal{F}_{\mathsf{lts}}^{\mathsf{bc}}$ ii. $\mathcal{F}_{\mathsf{lts}}^{\mathsf{ins,pki}}$ iii. $\mathcal{F}_{\mathsf{lts}}^{\mathsf{aut}}$

consist of the functions $f \in \mathcal{F}$ for which a constant round protocol $\pi \in \mathsf{Poly}$ that implements $\mathsf{F}_f^{\mathsf{ab}}$ with LT security in presence of an active adversary from

(i) an authenticated broadcast channel;

(ii) a complete network of insecure channels and a PKI;

(iii) a complete network of authenticated channels;

respectively. ◊

We now show that the classes defined in the previous section are all equivalent to $\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}}$. First, we observe that once we allow CO assumptions during the protocol execution, we can force semi-honest behavior (i.e. that the adversary behaves according to the protocol) using an unconditionally hiding commitment scheme and zero-knowledge arguments of knowledge:

Theorem 3.6.2. *If one-way functions (OWF) exist, we have* $\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} = \mathcal{F}_{\mathsf{lts}}^{\mathsf{bc}}$.

Proof of Theorem 3.6.2. By Lemma 2.8.3 we have $\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} = \mathcal{F}_{\mathsf{sh}}^{\mathsf{bc}}$, so it suffices to show $f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{bc}} \iff f \in \mathcal{F}_{\mathsf{lts}}^{\mathsf{bc}}$. We proceed to show each implication separately.

$f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{bc}} \implies f \in \mathcal{F}_{\mathsf{lts}}^{\mathsf{bc}}$**:** A long-term secure protocol is already by definition secure against unbounded semi-honest adversaries.

$f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{bc}} \implies f \in \mathcal{F}_{\mathsf{lts}}^{\mathsf{bc}}$**:** Let $f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{bc}}$. We show that $f \in \mathcal{F}_{\mathsf{lts}}^{\mathsf{bc}}$ by constructing a protocol π for computing f and proving its security.

The protocol π works as follows: First, we obtain an unconditionally hiding commitment scheme from the given OWF using [HR07]. Then the parties commit to their inputs and give a perfectly zero-knowledge argument of knowledge of their inputs. Now the parties execute the semi-honestly secure protocol $\widetilde{\pi}$ for function f, that exists as $f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{bc}}$, with the following modifications: After each computation step they commit to the result and give a perfect ZK argument that it was computed correctly. Instead of sending messages they now simply open commitments. If at any point a zero-knowledge argument fails, the protocol π aborts.

To show that this protocol is LT secure (security with abort), we construct an efficient simulator $\mathsf{S} \in \mathsf{Poly}$ such that for any efficient adversary $\mathsf{A} \in \mathsf{Poly}$ every distinguisher $\mathsf{D} \in \mathsf{Algo}$ has only negligible distinguishing advantage.

The simulator S feeds the input x_{A} received from the distinguisher D to the adversary A. Now the adversary A has to commit to his inputs $x_{\mathcal{A}}$ and give a zero-knowledge argument of knowledge of the inputs $x_{\mathcal{A}}$. By the definition of a proof of knowledge, there exists a knowledge-extractor W interacting with the adversary A that can extract the knowledge $x_{\mathcal{A}}$ of the adversary A. Since simulator S can execute multiple instances of the adversary A and branch its execution with the adversary A's randomness fixed, the simulator S can use the knowledge-extractor W to extract the committed inputs $x_{\mathcal{A}}$. Simulator S forwards the extracted inputs $x_{\mathcal{A}}$ to the ideal functionality $\mathsf{F}_f^{\mathsf{ab}}$ and in return receives the outputs $y_{\mathcal{A}}$. The simulator S then determines some $\tilde{x}_{\mathcal{H}}$ such that

$$f_{\mathcal{A}}(\tilde{x}_{\mathcal{H}}, x_{\mathcal{A}}) = y_{\mathcal{A}}$$

using the function table of f. Simulator S now proceeds to run the protocol $\pi_{\mathcal{H}}(\tilde{x}_{\mathcal{H}})$ for the honest parties with the adversary A. If the adversary A deviates from the protocol π (i.e. a zero-knowledge argument fails), simulator S causes functionality $\mathsf{F}_f^{\mathsf{ab}}$ to abort. In the end, simulator S forwards the output y_{A} of the adversary A to the distinguisher D.

We now have to show that this simulator S meets our requirements:

It is clear that simulator S is efficient, since adversary A, knowledge-extractor W and protocol π are efficient and finding the input $\tilde{x}_{\mathcal{H}}$ in the constant size function table can be done in constant time.

It remains to see that the real and the ideal settings are indistinguishable for any distinguisher D: Let $x_{\mathcal{H}}$ be the input of the distinguisher D

to the honest parties. Due to the way the input $\tilde{x}_{\mathcal{H}}$ is chosen by the simulator S the protocol execution between simulator S and adversary A will differ at most negligibly from the real protocol execution and therefore the output of the adversary differs only negligibly from the output in the real setting. This directly follows from the security of the underlying semi-honestly secure protocol $\widetilde{\pi}$.

Since the adversary A must be efficient, it cannot forge the zero-knowledge arguments with non-negligible probability. Thus, if the adversary A deviates from the protocol π, this results in a failed zero-knowledge argument with overwhelming probability and then the protocol aborts. So our use of zero-knowledge arguments actually forces semi-honest behavior (with abort).

Since we use a perfectly hiding bit commitment scheme and the arguments are perfect zero-knowledge, no additional information is flowing compared to the original protocol. Thus, we find that privacy cannot be violated anymore after protocol termination, even by unbounded adversaries. □

Theorem 3.6.3. *If one-way functions exist, the long-term securely computable (without fairness) functions under public-key infrastructure and insecure channels, or under public discussion or in the authenticated channels model are exactly the semi-honestly computable functions:*

$$\mathcal{F}_{\mathsf{lts}}^{\mathsf{ins,pki}} = \mathcal{F}_{\mathsf{lts}}^{\mathsf{bc}} = \mathcal{F}_{\mathsf{lts}}^{\mathsf{aut}} = \mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}}.$$

Proof. We prove this by showing the three inclusions $\mathcal{F}_{\mathsf{lts}}^{\mathsf{bc}} \subseteq \mathcal{F}_{\mathsf{lts}}^{\mathsf{ins,pki}}$, $\mathcal{F}_{\mathsf{lts}}^{\mathsf{ins,pki}} \subseteq \mathcal{F}_{\mathsf{lts}}^{\mathsf{aut}}$, and $\mathcal{F}_{\mathsf{lts}}^{\mathsf{aut}} \subseteq \mathcal{F}_{\mathsf{lts}}^{\mathsf{bc}}$. Theorem 3.6.3 then directly follows from Theorem 3.6.2.

The first inclusion $\mathcal{F}_{\mathsf{lts}}^{\mathsf{bc}} \subseteq \mathcal{F}_{\mathsf{lts}}^{\mathsf{ins,pki}}$ holds as we can use the Dolev-Strong-Protocol [DS83] to obtain authenticated BC in the PKI setting.

The second inclusion $\mathcal{F}_{\mathsf{lts}}^{\mathsf{ins,pki}} \subseteq \mathcal{F}_{\mathsf{lts}}^{\mathsf{aut}}$ holds as using detectable precomputation [FHHW03] we can establish a PKI in the authenticated channels model.[18]

The third inclusion $\mathcal{F}_{\mathsf{lts}}^{\mathsf{aut}} \subseteq \mathcal{F}_{\mathsf{lts}}^{\mathsf{bc}}$ holds as given authenticated BC, authenticated channels can be implemented by simply broadcasting messages and instructing honest parties other than the intended recipient to ignore the message. □

[18] Note that robustness is not required here: The establishment of the PKI may fail, but then the protocol simply aborts.

Theorem 3.6.3 is optimal in the sense that we cannot hope to implement all functions $f \in \mathcal{F}_{\mathsf{lts}}^{\mathsf{ins,pki}}$ with robustness or even fairness. Of course we have (by definition) robust LT (even IT) secure protocols for the functions $f \in \mathcal{F}_{\mathsf{act}}^{\mathsf{bc}}$. But e.g. the symmetric XOR function

$$f_{\mathrm{XOR}}(x_1, x_2) := (x_1 \,\mathrm{XOR}\, x_2, x_1 \,\mathrm{XOR}\, x_2)$$

is by the combinatorial characterizations of the previous sections

$$f_{\mathrm{XOR}} \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{bc}} \setminus \mathcal{F}_{\mathsf{act}}^{\mathsf{bc}} \subset \mathcal{F}_{\mathsf{lts}}^{\mathsf{bc}} \setminus \mathcal{F}_{\mathsf{act}}^{\mathsf{bc}}.$$

Now a fair implementation of f_{XOR} would clearly imply a fair cointoss, which by [Cle86] cannot be implemented in the model under consideration. As such the security without fairness as guaranteed by Theorem 3.6.3 is indeed the best we can hope for.

3.6.1 Long Term Security with designated Aborter

As mentioned above we cannot generally guarantee robustness or even fairness for a LT secure protocol π computing a function $f \in \mathcal{F}_{\mathsf{lts}}^{\mathsf{aut}}$. However, under stronger CO assumptions, we can guarantee that only a specific designated party can unfairly abort the protocol (after obtaining output and before the honest parties can generate output). This may be of practical relevance where a specific party is not trusted, but can be relied upon not to abort the protocol. For instance a party may have a vested interest in the successful termination of the protocol regardless of the outcome. One may think of an auctioneer that gets paid only if the auction terminates successfully. Or a party may act in an official capacity and cannot abort the protocol for legal reasons.

We will show that stronger guarantees of this type are obtainable if the underlying CO assumption allows for an oblivious transfer (OT) protocol which is LT secure against one of the participants. Enhanced trapdoor one-way permutations are an example of such an assumption [Gol04]. It is generally believed that OT is *not* implied by OWFs, meaning that LT security with designated aborter appears to require strictly stronger assumptions than plain LT security.

Lemma 3.6.4. *Any semi-honestly computable function $f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} = \mathcal{F}_{\mathsf{lts}}^{\mathsf{ins,pki}}$ can be computed using a protocol π which is long-term secure with designated aborter (LTS-DA), i.e. implements functionality $\mathsf{F}_f^{\mathsf{des}}$ with CO security and simultaneously functionality $\mathsf{F}_f^{\mathsf{ab}}$ with LT security in the insecure channels model*

with PKI iff computationally secure oblivious transfer LT-secure against one participant (CO-OT+) exists.

A proof of this lemma is given below. Essentially we apply the protocol compiler of [GMW87, Gol04] to the distributed circuit describing the semi-honestly secure protocol for function f in such a fashion that gates owned by a specific party P_i are computed with computational primitives that IT protect P_i. Reconstruction is in the end done toward the designated party P_1, which then ensures that the remaining parties can reconstruct. As a result the protocol is computationally correct, and information-theoretically no one learns more than in the semi-honestly secure protocol for function f.

Proof of Lemma 3.6.4. Let $f \in \mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}}$ be a semi-honestly computable function. From Theorem 3.3.6 we obtain a semi-honestly secure protocol $\widetilde{\pi}$. We "pad" the protocol $\widetilde{\pi}$ with dummy-messages, such that the number of rounds does not depend on the inputs anymore. We view the resulting protocol as a distributed circuit, where each gates is owned by a party P_i, depending on which party executes a specific computation. Communication is then nothing else than a wire between gates owned by different parties.

From this distributed circuit we can obtain a CO secure protocol for the function f using the Goldreich compiler [Gol04], that (CO) securely computes any circuit using a CO-OT+ functionality.

As in the case of plain LT security each party first commits to their input using unconditionally hiding bit-commitments and proofs knowledge of the input using a perfectly zero-knowledge argument of knowledge. In the following, since we are interested in an LT secure protocol π for f with designated aborter P_1, we need to take special care when applying the Goldreich compiler. Concretely, we only use unconditionally hiding bit-commitments and perfect zero-knowledge arguments and we take care to apply the CO-OT+ in such a fashion that it is LT secure for P_i when computing a gate owned by P_i. When the protocol is complete, the result is first opened towards P_1, who then ensures that the result is opened to the remaining parties.

As shown in [Gol04] the protocol π computes function f CO securely and the designated abort property is provided via the opening procedure, since, as Goldreich shows, no party learns anything until the opening phase. It remains to argue LT security.

In protocol π the structure of the underlying semi-honestly IT secure protocol $\widetilde{\pi}$ is preserved. Due to the way we employ CO-OT+, information-theoretically, each party only receives information about inputs and outputs of gates it computes itself in protocol $\widetilde{\pi}$. Therefore no party receives more information than in protocol $\widetilde{\pi}$, and after termination of the protocol security, even against unbounded distinguishers, is guaranteed.

Simulators as demanded by the definition of security are easily constructed from the above. The proofs of knowledge in the beginning of the protocol are used to extract an input. A simulator passes this input to the ideal functionality and obtain output from the ideal functionality. The simulator then determines an input of the honest party that is consistent with the input extracted from the adversary and the output received from ideal functionality. Using this input the simulator can simulate a regular protocol execution toward the adversary and use the output of the adversary as its own. Indistinguishability of such a simulation follows once again directly from the above arguments. □

3.7 Classification of 2-party Functions

Combining the results of this work and of [KMQ08], we can derive a complete combinatorial classification of the 2-party functions $\mathcal{F}_2$ by completeness and computability.

First note that functions equivalent under the relation renaming as discussed in Definition 3.4.2 or [KMQ08] are mutually locally reducible under all security paradigms considered in this work. In particular for equivalence under renaming we have

$$f^{(1)} \equiv f^{(2)} \quad \Longrightarrow \quad \mathsf{F}_{f^{(1)}} \succcurlyeq^{\mathsf{PFE}}_{\mathsf{act}} \mathsf{F}_{f^{(2)}} \succcurlyeq^{\mathsf{PFE}}_{\mathsf{act}} \mathsf{F}_{f^{(1)}} \quad \text{and} \quad \mathsf{F}_{f^{(1)}} \succcurlyeq^{\mathsf{PFE}}_{\mathsf{pas}} \mathsf{F}_{f^{(2)}} \succcurlyeq^{\mathsf{PFE}}_{\mathsf{pas}} \mathsf{F}_{f^{(1)}}.$$

Next we define an equivalence relation *matching* on the set of classes $\mathcal{F}_2/\equiv$ induced by the relation renaming (and thereby on all functions $\mathcal{F}_2$) by isolating inputs that lead to identical behavior and regarding functions as matching if, after eliminating such trivially redundant inputs, they are renamings:

Definition 3.7.1. Given a 2-party function $f \in \mathcal{F}_2$ we say x_A *matches* x'_A for inputs $x_A, x'_A \in \mathcal{X}_A$, iff x_A dominates x'_A and x'_A dominates x_A. The

matching relation is an equivalence relation on $\mathcal{X}_A$. By $\bar{\mathcal{X}}_A$ we designate a set of representatives. $\bar{\mathcal{X}}_B$ is defined analogously.

We then call $\bar{f} := f|_{\bar{\mathcal{X}}_A \times \bar{\mathcal{X}}_B}$ the *weakly redundancy-free version* of f and for $f^{(1)}, f^{(2)} \in \mathcal{F}_2$ we write $f^{(1)} \cong f^{(2)}$ if $\bar{f}^{(1)} \equiv \bar{f}^{(2)}$ Furthermore, for $x_A \in \mathcal{X}_A$ and $x_B \in \mathcal{X}_B$ let $\bar{x}_A \in \bar{\mathcal{X}}_A$ and $\bar{x}_B \in \bar{\mathcal{X}}_B$ be the (unique) elements that match x_A and x_B respectively. $\Diamond$

Like the redundancy-free version $\hat{f}$ of f, the weakly redundancy-free version $\bar{f}$ of f is well defined up to renaming.

Before we can state the actual classification, we have to reiterate another result of [KMQ08]:

Theorem 3.7.2 (Complete Functions [KMQ08])**.** *The classes $\mathcal{C}_{2\mathsf{act}}$, $\mathcal{C}_{2\mathsf{sh}}$ and $\mathcal{C}_{2\mathsf{pas}}$ of actively, semi-honestly, and passively complete 2-party functions are the classes of functions $f \in \mathcal{F}_2$ to which all other 2-party functions can be securely reduced in presence of an active, semi-honest or passive adversary respectively.*

The classes $\mathcal{C}_{2\mathsf{act}} = \mathcal{C}_{2\mathsf{sh}}$ consist of exactly the functions $f \in \mathcal{F}_2$ where $\hat{f} \in \mathcal{C}_{2\mathsf{pas}}$. The class $\mathcal{C}_{2\mathsf{pas}}$ consists of exactly the functions $f \in \mathcal{F}_2$ where

$$\begin{aligned}&\exists\, a_1, a_2 \in \mathcal{X}_A,\ b_1, b_2 \in \mathcal{X}_B :\\ &\quad f_A(a_1, b_1) = f_A(a_1, b_2) \ \wedge\ f_B(a_1, b_1) = f_B(a_2, b_1) \wedge\\ &\quad (\ f_A(a_2, b_1) \neq f_A(a_2, b_2) \ \vee f_B(a_1, b_2) \neq f_B(a_2, b_2)\ ).\end{aligned}$$

We refer to this combinatorial structure as minimal OT.

Note that $f \in \mathcal{C}_{2\mathsf{pas}}$ iff $f \in \mathcal{C}_{2\mathsf{act}}$ or $\hat{f} \not\equiv \bar{f}$. This is clear from Kraschewki's result as stated above and from the observation that $\hat{f} \not\equiv \bar{f}$ implies a minimal OT. We then arrive at the following

Theorem 3.7.3 (Classification)**.** *The class of 2-party functions is a disjoint union of three sets*

$$\begin{aligned}\mathcal{F}_2 &= \mathcal{C}_{2\mathsf{act}} \cup \mathcal{F}_{2\mathsf{act}} \cup \mathcal{F}^{\mathsf{nct}}_{2\mathsf{act}} && \textit{or}\\ &= \mathcal{C}_{2\mathsf{sh}} \cup \mathcal{F}_{2\mathsf{sh}} \cup \mathcal{F}^{\mathsf{nct}}_{2\mathsf{sh}} && \textit{or}\\ &= \mathcal{C}_{2\mathsf{pas}} \cup \mathcal{F}_{2\mathsf{pas}} \cup \mathcal{F}^{\mathsf{nct}}_{2\mathsf{pas}}\end{aligned}$$

where nct *stand for "neither complete nor computable". Now*

$$\begin{aligned}&\emptyset \neq \mathcal{F}_{2\mathsf{act}}, \mathcal{F}_{2\mathsf{pas}} \subsetneq \mathcal{F}_{2\mathsf{act}} \cup \mathcal{F}_{2\mathsf{pas}} \subsetneq \mathcal{F}_{2\mathsf{sh}}\\ &\emptyset \neq \mathcal{F}^{\mathsf{nct}}_{2\mathsf{pas}} \subsetneq \mathcal{F}^{\mathsf{nct}}_{2\mathsf{sh}} \subsetneq \mathcal{F}^{\mathsf{nct}}_{2\mathsf{act}}\\ &\emptyset \neq \mathcal{C}_{2\mathsf{act}} = \mathcal{C}_{2\mathsf{sh}} \subsetneq \mathcal{C}_{2\mathsf{pas}}\end{aligned}$$

More precisely,

$$\begin{aligned}
\mathcal{F}_{\mathsf{2act}} \setminus \mathcal{F}_{\mathsf{2pas}} &= \{f \in \mathcal{F}_{\mathsf{2act}} \mid \bar{f} \not\equiv \hat{f}\} \\
\mathcal{F}_{\mathsf{2sh}} \setminus \mathcal{F}_{\mathsf{2pas}} &= \{f \in \mathcal{F}_{\mathsf{2sh}} \mid \bar{f} \not\equiv \hat{f}\} \\
\mathcal{F}^{\mathsf{nct}}_{\mathsf{2pas}} &= \{f \in \mathcal{F}^{\mathsf{nct}}_{\mathsf{2sh}} \mid \bar{f} \equiv \hat{f}\} \\
\mathcal{F}^{\mathsf{nct}}_{\mathsf{2act}} &= \mathcal{F}^{\mathsf{nct}}_{\mathsf{2sh}} \cup (\mathcal{F}_{\mathsf{2sh}} \setminus \mathcal{F}_{\mathsf{2act}}) \\
\mathcal{C}_{\mathsf{2pas}} &= \mathcal{C}_{\mathsf{2act}} \cup \{f \in \mathcal{F}_2 \mid \hat{f} \not\equiv \bar{f}\}.
\end{aligned}$$

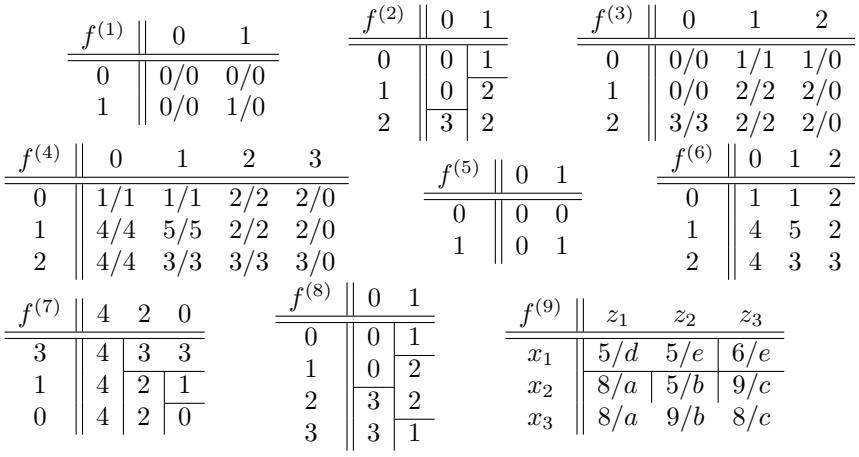
The above results are directly derived from the combinatorial descriptions of the function classes that can be found in the preceding sections and, as far as complete functions are concerned, in [KMQ08].

Using the examples in Fig. 3.2 we can illustrate that the above inclusions are indeed proper:

$$\begin{aligned}
f^{(1)} &\in \mathcal{F}_{\mathsf{2act}} \setminus \mathcal{F}_{\mathsf{2pas}} \subsetneq \mathcal{C}_{\mathsf{2pas}} \setminus \mathcal{C}_{\mathsf{2act}} \\
f^{(2)} &\in \mathcal{F}_{\mathsf{2pas}} \setminus \mathcal{F}_{\mathsf{2act}} \subsetneq \mathcal{F}^{\mathsf{nct}}_{\mathsf{2act}} \setminus \mathcal{F}^{\mathsf{nct}}_{\mathsf{2sh}} \\
f^{(3)} &\in \mathcal{F}_{\mathsf{2sh}} \setminus \mathcal{F}_{\mathsf{2act}} \cup \mathcal{F}_{\mathsf{2pas}} \\
f^{(4)} &\in \mathcal{C}_{\mathsf{2pas}} \cap \mathcal{F}^{\mathsf{nct}}_{\mathsf{2sh}} \subsetneq \mathcal{C}_{\mathsf{2pas}} \setminus \mathcal{C}_{\mathsf{2act}}.
\end{aligned}$$

3.8 Conclusions

We provided a combinatorial characterization of the classes $\mathcal{F}^{\mathsf{aut}}_{\mathsf{pas}}$, $\mathcal{F}^{\mathsf{aut}}_{\mathsf{sh}}$ and $\mathcal{F}^{\mathsf{aut}}_{\mathsf{act}}$ of n-party functions IT securely computable in the authenticated channels model (with broadcast) in presence of active, semi-honest or passive adversaries corrupting an arbitrary number of parties. From our findings we derived a characterization of the functions computable with long-term (LT) security, where we assume that the adversary is computationally bounded *during* the execution of the protocol *only*. That is, we rely on CO assumptions, but only for the duration of the protocol execution; thereafter, a failure of the CO assumptions must not compromise security. We characterize the class $\mathcal{F}^{\mathsf{ins},\mathsf{pki}}_{\mathsf{lts}}$ of functions that can be computed LT securely in an internet-like setting, where a complete network of insecure channels and a PKI are available. Our results are constructive in that, for every function proven computable in a given setting, we give a secure protocol.



$f^{(1)}$	0	1
0	0/0	0/0
1	0/0	1/0

$f^{(2)}$	0	1
0	0	1
1	0	2
2	3	2

$f^{(3)}$	0	1	2
0	0/0	1/1	1/0
1	0/0	2/2	2/0
2	3/3	2/2	2/0

$f^{(4)}$	0	1	2	3
0	1/1	1/1	2/2	2/0
1	4/4	5/5	2/2	2/0
2	4/4	3/3	3/3	3/0

$f^{(5)}$	0	1
0	0	0
1	0	1

$f^{(6)}$	0	1	2
0	1	1	2
1	4	5	2
2	4	3	3

$f^{(7)}$	4	2	0
3	4	3	3
1	4	2	1
0	4	2	0

$f^{(8)}$	0	1
0	0	1
1	0	2
2	3	2
3	3	1

$f^{(9)}$	z_1	z_2	z_3
x_1	$5/d$	$5/e$	$6/e$
x_2	$8/a$	$5/b$	$9/c$
x_3	$8/a$	$9/b$	$8/c$

Figure 3.2: Examples. Inputs for A are shown to the right, inputs for B on top. For asymmetric functions, outputs are denoted y_A/y_B; for symmetric functions only the common output of both parties is listed.

More precisely, we showed that semi-honest computability and LT secure computability amount to the same, i.e. $\mathcal{F}_{\mathsf{sh}}^{\mathsf{aut}} = \mathcal{F}_{\mathsf{lts}}^{\mathsf{bc}} = \mathcal{F}_{\mathsf{lts}}^{\mathsf{aut}} = \mathcal{F}_{\mathsf{lts}}^{\mathsf{ins,pki}}$, where the classes $\mathcal{F}_{\mathsf{lts}}^{\mathsf{aut}}$ and $\mathcal{F}_{\mathsf{lts}}^{\mathsf{bc}}$ are defined analogously to $\mathcal{F}_{\mathsf{lts}}^{\mathsf{ins,pki}}$, but rely on a network of authenticated channels or authenticated broadcast respectively as communication resources. Furthermore, we show that the class of actively computable functions $\mathcal{F}_{\mathsf{act}}^{\mathsf{aut}} \subsetneq \mathcal{F}_{\mathsf{lts}}^{\mathsf{ins,pki}}$ is a proper subset of the class of LT securely computable functions $\mathcal{F}_{\mathsf{lts}}^{\mathsf{ins,pki}}$, demonstrating the usefulness of the notion long-term security.

As the actively computable functions in the class $\mathcal{F}_{\mathsf{act}}^{\mathsf{aut}}$ are robustly (and therefore fairly) computable, our results on active computability can be interpreted along the lines Gordon et al. [GHKL08], who discuss the fair computability of binary 2-party functions in the computational setting. Our results apply to the IT scenario instead of the CO setting, there however, our results are much more general in that they pertain to arbitrary n-party functions. We showed that for the functions $\mathcal{F}_{\mathsf{lts}}^{\mathsf{ins,pki}}$ fairness is generally not achievable. However, for the functions $\mathcal{F}_{\mathsf{lts}}^{\mathsf{ins,pki}}$ we can guarantee LT security with designated aborter, where only a specific designated party can prematurely abort the protocol after having learned the output. Astonishingly, CO secure oblivious transfer (OT) is used in our

construction, even though OT itself cannot be realized with long-term security.

We remark, that from a practical point of view, LT security is a useful notion if we deal with sensitive data that has to remain private beyond a limited time frame in a setting where a majority of the parties may be corrupted. In such a setting general IT secure SFE protocols like [BGW88] fail, as they do not tolerate corrupted majorities. CO secure protocols can tolerate corrupted majorities (if fairness is not required) but, as time passes, progress in hardware or algorithms may invalidate our computational assumptions and jeopardize the privacy of our computation. As the problem with computational assumptions is not so much that these could be unjustified right now, but rather their possible future invalidation, LT security is a viable alternative to IT security in this case. And indeed we could show that $\mathcal{F}_{\mathsf{act}}^{\mathsf{aut}} \subsetneq \mathcal{F}_{\mathsf{lts}}^{\mathsf{ins},\mathsf{pki}}$, i.e. there are functions that cannot be computed with IT security in presence of dishonest majorities, but can be computed with LT security.

Finally, collecting results form the literature, especially [Kus92, KMQ08], and adding the results of this work, we obtain a complete taxonomy of 2-party functions by computability and completeness in the IT setting.

Chapter 4

On Optimally Hybrid-Secure MPC

4.1 Introduction

In this chapter we focus on devising general MPC protocols that provide optimal graceful degradation of security properties as the number t of corrupted parties increases. The material presented in this chapter is published in [LRM09].

Most general MPC protocols in the literature are designed to be secure either against IT or against CO adversaries. We know from the previous chapter and the literature that we can only hope to achieve fairness or IT security in presence of corrupted majorities for a very restricted class of functions. In the following we return to the problem of general MPC. That is, we are interested in protocols suitable for implementing any arbitrary functionality. As such IT security or fairness in presence of arbitrarily many corrupted parties are out of reach.

IT secure MPC protocols then have the disadvantage that only a corrupted minority can be tolerated without compromising security. On the other hand, CO protocols can tolerate any number of corrupted parties if robustness and fairness are not required, but they are based on unproven intractability assumptions. Invalidation of the underlying assumptions generally leads to a complete loss of security, even if only a single party is corrupted.

In contrast, our MPC protocols with hybrid security allow for graceful degradation of security, from full IT security in case of few corruptions, to CO security without fairness in case of a corrupted majority.

4.1.1 Secure Multi-Party Computation

A first general solution to the MPC problem was given by [GMW87], based on computational (CO) intractability assumptions and a broadcast (BC) channel. They achieve security with robustness against $t < \frac{n}{2}$ actively corrupted parties or with agreement on abort against $t < n$ actively corrupted parties as described in [Gol04]. On the other hand [BGW88], and independently [CCD88], presented protocols which are information-theoretically (IT) secure and require no BC channel. However, they prove that security can only be achieved as long as $t < \frac{n}{3}$ parties are corrupted. When no robustness is required (detectable MPC) [FHHW03] or if a BC channel is available [RB89], this bound can be improved to $t < \frac{n}{2}$.

MPC is generally treated in a setting where parties are connected by a complete and synchronous network of secure channels. Additionally an authenticated synchronous BC channel or a public key infrastructure (PKI) may be available. Universally composable (UC) MPC protocols usually also require a common reference string (CRS) [Can01, CF01]. Unless otherwise stated we assume a CRS and a complete and synchronous network Net^n of secure channels and an authenticated broadcast channel BC^n in this chapter.

4.1.2 Hybrid Security

Most MPC protocols are designed to be secure either against IT or against CO adversaries. IT MPC protocols have the disadvantage that only a corrupted minority can be tolerated without compromising security. On the other hand, CO protocols can tolerate any number of corrupted parties if robustness and fairness are not required, but they are based on unproven intractability assumptions. Invalidation of the underlying assumptions generally leads to a complete loss of security, even if only a single party is corrupted.

MPC protocols with hybrid security provide different levels of security, depending on the number of corrupted parties. Thus they allow for

graceful degradation of security. Specifically, we discuss a protocol offering IT security in case of few corruptions, but still CO security in case of many corruptions.

4.1.3 Contributions and Related Work

We provide UC-secure MPC protocols that combine IT and CO security and allow for flexible trade-offs between security with robustness, fairness, or agreement on abort. For any robustness parameter $\rho < \frac{n}{2}$ we describe an MPC protocol π^ρ that, given a CRS, simultaneously provides IT security with robustness against static adversaries actively corrupting $t \leq \rho$ parties, IT security with fairness (no robustness) for $t < \frac{n}{2}$, and CO security with agreement on abort (no fairness) for $t < n - \rho$. Furthermore, we provide proofs in the UC model as well as the stand-alone model without CRS.

In [Cha89] Chaum sketches a construction aimed at guaranteeing CO privacy for any number of actively corrupted parties, and simultaneously IT privacy, given that only a minority of the parties is corrupted. In contrast to our work, [Cha89] does not discuss fairness, or robustness. Furthermore, several critical details are neglected. In fact, correctness is not guaranteed in [Cha89]. A central element of both Chaum's approach and ours is the emulation of a player in one protocol, using another protocol. In [HM00], this technique was discussed in the stand-alone setting for perfectly secure MPC and applied to general adversary structures. We contribute a formal treatment of this technique in the UC setting.

Fitzi et al. [FHW04] also combine IT and CO security: Up to a first threshold t_p, the security is IT. Between t_p and the second threshold t_σ IT security is guaranteed conditioned on the consistency of the underlying PKI. Finally, between t_σ and T the protocol is as secure as the signature scheme in use. Fitzi et al. show that their notion of hybrid MPC is achievable if and only if $(2T + t_p < n) \wedge (T + 2t_\sigma < n)$, which they prove to be tight.

Another work by Fitzi et al. [FHHW03] improves upon [BGW88, CCD88] in the IT setting, when no BC channel is available by allowing for two thresholds t_v and t_c where $t_v = 0$ or $t_v + 2t_c < n$. Then for $t \leq t_v$ corrupted parties, fully secure MPC is achieved while for $t_v < t \leq t_c$ corrupted parties non-robust (but fair) MPC is accomplished.

Both [FHW04] and [FHHW03] largely focus on a setting without BC channel. When a BC channel is provided our results improve substantially upon those of [FHW04, FHHW03]. As [FHHW03] focuses exclusively on IT MPC and [FHW04] only treats robust MPC, both [FHW04, FHHW03] do not reach beyond $t < \frac{n}{2}$ corrupted parties, nor are they easily extended, whereas we can guarantee CO security with agreement on abort for $t < n - \rho$. In the setting without BC channel, and for $\rho > 0$ our results match those of [FHHW03] (which they prove optimal for this case). However, for the special case that $\rho = 0$ (i.e., no robustness is required) our construction achieves IT fairness for $t < \frac{n}{2}$, and CO security with abort for $t < n$ corrupted parties, which goes beyond [FHHW03].

In [IKLP06] and [Kat06] trade-offs between robust and non-robust MPC are discussed, but only in the CO setting. They show that a protocol which guarantees robustness for up to ρ corrupted players can be secure with abort against at most $n - \rho$ corrupted players, and give CO secure protocols that match these bounds. Our protocol π^ρ is optimal under the bounds of [IKLP06, Kat06] but beyond that also provides IT security for $t < \frac{n}{2}$. Furthermore, we match the bound $t < \frac{n}{2}$ on fairness for general MPC put forth in [Cle86], and the bound $t < \frac{n}{2}$ on IT security in the presence of active adversaries (e.g. [Kil00]).

So, in conclusion we obtain a flexible and optimal trade-off between IT robustness, IT fairness and CO security with agreement on abort, and give proofs of these properties in the UC setting. Also, on the technical side, we contribute a treatment of player emulation in the UC setting.

4.2 Conventions and Simplifications

In the following we will be interested in implementing an arbitrary n-party functionality F with hybrid security. The only restrictions on functionality F are that it provides an I/O-interface to each of the n parties and that it notifies the adversary of the length of any input or output, and the identity of the sending or receiving party P_i.

4.2.1 Secure Implementation

We discuss hybrid-secure protocols that provide different security properties depending on the number of corrupted parties and on the computational setting. As such we will use corruption and computational

model aware functionalities that exhibit different behavior depending on the number t of corrupted parties and on the computational setting (CO or IT). We will say that a protocol π (UC or stand-alone) securely implements an ideal functionality F if π securely implements F in both the CO and the IT setting. We generally only consider efficient simulators, since otherwise, IT security does not imply CO security.

4.2.2 Symmetric vs. Asymmetric Functionalities

For simplicity, we assume that functionality F is symmetric, i.e., when functionality F makes output, it provides the same output y to *all* participants. This is without loss of generality for the purposes of this chapter because in this chapter we consider general MPC:

As shown in [CDG88], securely implementing an asymmetric functionality F providing a separate output for each party, can IT securely be reduced to securely implementing a symmetric functionality F':

In addition to the usual inputs of functionality F the functionality F' requests ℓ_i random bits r_i from each party P_i (where ℓ_i is the length of the output y_i). Instead of $y = (y_1, \ldots, y_n)$, F' then yields $y' = (y_1 \oplus r_1, \ldots, y_n \oplus r_n)$ and provides the same output y' to all parties.

Each party can recover its own output as $y_i = y'_i \oplus r_i$. It follows from the security of one-time-pad (OTP) encryption that no party is able to obtain any information about the other parties' output, even with unbounded computational resources, given that the input remains private. This idea was presented in [GMW87] for computational security with asymmetric encryption. The generalization to the IT case with an OTP encryption was discussed e.g. briefly in [CDG88] (there called *private output*).

Thus, for the purpose of this chapter we may restrict our attention to symmetric n-party functionalities F.

4.3 Hybrid-secure MPC: The Protocol π^{ρ}

We present a protocol π^{ρ} that UC securely implements hybrid-secure MPC from a common reference string CRS and a complete n-party network Net^n (consisting of secure channels and an authenticated n-party broadcast channel BC^n). Furthermore, we provide a stand-alone secure

protocol π^{ρ}_{SA} which provides the same guarantees as protocol π^{ρ} without relying on a CRS. More precisely, given a robustness parameter $\rho < \frac{n}{2}$ and an n-party functionality F, protocol π^{ρ} implements F simultaneously providing IT full security in the presence of up to $t \leq \rho$ actively corrupted parties, IT fair security (no robustness) for $t < \frac{n}{2}$ and CO abort security (no fairness) for $t < n - \rho$. These security requirements are formalized via the ideal functionality F^{ρ} in Fig. 4.1.

We construct protocol π^{ρ} in three steps:

1. We show how to IT securely emulate and integrate an additional party P_0 given an n-party network Net^n (see Section 4.4). This amounts to emulating the protocol π_0 of party P_0 and an $(n+1)$-party network Net^{n+1}.

2. We exhibit an $(n+1)$-party MPC protocol $\pi^{\mathsf{des},\rho}$ (see Section 4.5) that has a *designated party property*: Protocol $\pi^{\mathsf{des},\rho}$ is run among the n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$, and a special, designated party P_0. Fairness and robustness of protocol $\pi^{\mathsf{des},\rho}$ depend centrally on the honesty of the designated party P_0. Furthermore, the designated party P_0 has IT privacy and correctness guarantees. In contrast, the parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ have only CO privacy and correctness guarantees. Such a designated party protocol $\pi^{\mathsf{des},\rho}$ can be obtained by modifying the protocol of [CLOS02] as described in Section 4.5.

   The strong security guarantees of protocol $\pi^{\mathsf{des},\rho}$ for the designated party P_0 are then transferred to the remaining parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ by having them emulate P_0: As long as the emulation is secure (for $t < \frac{n}{2}$), the emulated party P_0 can be regarded as honest and the resulting protocol will have the strong fairness, robustness, and correctness properties which protocol $\pi^{\mathsf{des},\rho}$ exhibits if the designated party P_0 is honest.

3. We provide an input protocol π^{in} (see Section 4.6), that transforms a designated party MPC into a hybrid-secure MPC, exploiting the designated party property.

By the UC theorem, these three steps can be aggregated into a protocol π^{ρ} that securely realizes hybrid-secure MPC as formalized by functionality F^{ρ}. Protocol π^{ρ} relies on the basic resources needed for the construction above, namely a CRS and an n-party network Net^n.[19] An

[19] Note that our construction uses multiple instances of the network Net^n. However, it is easy to securely implement multiple instances of Net^n from a single instance of Net^n by multiplexing.

overview of our construction can be found in Fig. 4.2.

Functionality F^ρ behaves as specified by the following table. That is, for a given computational setting (CO or IT) and number t of corrupted parties functionality F^ρ behaves exactly like the corresponding functionality listed under behavior below.

Adversarial Power		Behavior	
$t \leq \rho$	CO/IT	F	(full security)
$\rho < t < \frac{n}{2}$	CO/IT	$\mathsf{F}^{\mathsf{fair}}$	(fair security)
$\frac{n}{2} \leq t < n - \rho$	CO	F^{ab}	(abort security)
	IT	$\mathsf{F}^{\mathsf{noSec}}$	(no guarantees)
$n - \rho \leq t$	CO/IT	$\mathsf{F}^{\mathsf{noSec}}$	(no guarantees)

Figure 4.1: The ideal functionality F^ρ.

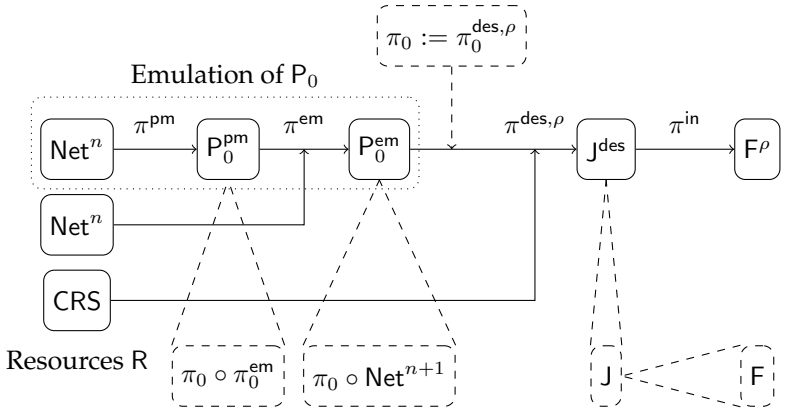


Figure 4.2: Protocol π^ρ: The construction of the hybrid MPC functionality F^ρ. Protocol π^ρ is the composition of the protocols π^{pm}, π^{em}, $\pi^{\mathsf{des},\rho}$, and π^{in} in the figure. The dashed boxes beneath the ideal functionalities contain hints on their behavior.

Theorem 4.3.1 (UC Security of π^ρ)**.** *Let* F *be an ideal* n*-party functionality and let* $\rho < \frac{n}{2}$ *be a robustness parameter. Let a* CRS *setup and a network* Net^n *(encompassing a complete and synchronous network of secure channels and an authenticated BC channel) be given. Protocol* π^ρ *then implements functional-*

ity F^{ρ} *UC securely against static adversaries corrupting any number* t *of parties. That is,* π^{ρ} *implements the ideal functionality* F

1. *with IT full security, given that* $t \leq \rho$,
2. *with IT fair security (as formalized by* $\mathsf{F}^{\mathsf{fair}}$*), given that* $t < \frac{n}{2}$*, and*
3. *with CO abort security (as formalized by* F^{ab}*), given that* $t < n - \rho$.

Proof. The UC security of protocol $\pi^{\rho} = \pi^{\mathsf{in}} \circ \pi^{\mathsf{des},\rho} \circ \pi^{\mathsf{em}} \circ \pi^{\mathsf{pm}}$ as claimed in Theorem 4.3.1 follows by the UC theorem from the UC security of its subprotocols π^{in}, $\pi^{\mathsf{des},\rho}$, π^{em}, and π^{pm}, discussed in Sections 4.6, 4.5, and 4.4. □

Note that our protocol does not tolerate an adversarially chosen CRS, even for $t \leq \rho$.[20] However, in our protocol π^{ρ} the CRS is only needed for the perfectly hiding or perfectly binding UC secure commitments of [DN02]. Thus, in the commitment hybrid model we achieve the same security guarantees as above without a CRS. CO assumptions sufficient for implementing the necessary CO primitives for protocol π^{ρ} include the p-subgroup assumption or the decisional composite residuosity assumption [DN02].

Furthermore, we can obtain a stand-alone secure variation π^{ρ}_{SA} of protocol π^{ρ} which stand-alone securely implements F^{ρ} *without* reliance on a CRS.

Theorem 4.3.2 (SA Security of π^{ρ}_{SA})**.** *Let* F *be an ideal* n*-party functionality and let* $\rho < \frac{n}{2}$ *be a robustness parameter. Let a network* Net^n *(encompassing a complete and synchronous network of secure channels and an authenticated BC channel) be given. Protocol* π^{ρ}_{SA} *then implements functionality* F^{ρ} *stand-alone securely against static adversaries corrupting any number* t *of parties. That is,* π^{ρ}_{SA} *implements the ideal functionality* F

1. *with IT full security, given that* $t \leq \rho$,
2. *with IT fair security (as formalized by* $\mathsf{F}^{\mathsf{fair}}$*), given that* $t < \frac{n}{2}$*, and*
3. *with CO abort security (as formalized by* F^{ab}*), given that* $t < n - \rho$.

[20] It is possible to minimize the reliance on the CRS such that our protocols tolerate an adversarially chosen CRS for $t \leq \rho$ by applying techniques from [GK08, GO07] and a $(t, 2t-1)$-combiner for commitments (e.g. [Her05]). However, this construction is beyond the scope of this work.

Protocol π^{ρ}_{SA} is constructed along the same lines as protocol π^{ρ} but relies on a stand-alone secure subprotocol $\pi^{\mathsf{des},\rho}_{\mathsf{SA}}$ instead of the UC secure subprotocol $\pi^{\mathsf{des},\rho}$. Protocols $\pi^{\mathsf{des},\rho}_{\mathsf{SA}}$ and $\pi^{\mathsf{des},\rho}$ are both discussed in Section 4.5.

The stand-alone security of protocol $\pi^{\rho}_{\mathsf{SA}} = \pi^{\mathsf{in}} \circ \pi^{\mathsf{des},\rho}_{\mathsf{SA}} \circ \pi^{\mathsf{em}} \circ \pi^{\mathsf{pm}}$ as claimed in Theorem 4.3.2 is proven in Section 4.7.

4.4 Emulating a Party

We now describe how to IT securely emulate a party P_0 and its network connection from an n-party network Net^n. Emulating party P_0 amounts to emulating the protocol machine π_0 which P_0 is supposed to run. Running a given emulated party protocol π_0 on an $(n+1)$-party network Net^{n+1} is formalized by means of the emulation functionality $\mathsf{P}^{\mathsf{em}}_0$ described next.

4.4.1 The Emulation Functionality $\mathsf{P}^{\mathsf{em}}_0$

Functionality $\mathsf{P}^{\mathsf{em}}_0$ (Fig. 4.3) formalizes a given emulated party protocol π_0 connected to an $(n+1)$-party network Net^{n+1}. Here, π_0 may be an arbitrary $(n+1)$-party MPC protocol machine with a communication interface connecting to Net^{n+1} and n I/O-interfaces corresponding to the emulating parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$. Functionality $\mathsf{P}^{\mathsf{em}}_0$ will then run $\mathsf{P}^{\mathsf{em}}_0 = \pi_0 \circ \mathsf{Net}^{n+1}$ for $t < \frac{n}{2}$. For $t \geq \frac{n}{2}$, functionality $\mathsf{P}^{\mathsf{em}}_0$ only formalizes a network Net^{n+1}, in short $\mathsf{P}^{\mathsf{em}}_0 = \mathsf{Net}^{n+1}$ for $t < \frac{n}{2}$. We may think of the emulated party P_0 running π_0 as being honest for $t < \frac{n}{2}$ and corrupted for $t \geq \frac{n}{2}$. This is optimal as IT fully secure general MPC is only achievable for $t < \frac{n}{2}$ corrupted parties [Cle86].

For $t < \frac{n}{2}$, protocol machine π_0 is connected to Net^{n+1} via its communication interface. The remaining interfaces (the n I/O-interfaces of protocol π_0 and the n open interfaces to Net^{n+1}) constitute the interfaces of functionality $\mathsf{P}^{\mathsf{em}}_0$: Functionality $\mathsf{P}^{\mathsf{em}}_0$ gives each party P_i ($i \in [n]$) access to one I/O-interface to π_0 and one interface to Net^{n+1}. For $t \geq \frac{n}{2}$, we have $\mathsf{P}^{\mathsf{em}}_0 = \mathsf{Net}^{n+1}$ and we consider the emulated party P_0 which is supposed to run protocol machine π_0 as corrupted. Accordingly, the interface of Net^{n+1} connected to π_0 for $t < \frac{n}{2}$ is exposed to the adversary for $t \geq \frac{n}{2}$.

Adversarial Power		Behavior	
$t < \frac{n}{2}$	CO/IT	$\pi_0 \circ \mathsf{Net}^{n+1}$	(emulation and network)
$\frac{n}{2} \leq t$	CO/IT	Net^{n+1}	($(n+1)$-network)

Figure 4.3: The ideal functionality $\mathsf{P}_0^{\mathsf{em}}$.

4.4.2 Implementing Functionality $\mathsf{P}_0^{\mathsf{em}}$

As a first step towards implementing the emulation functionality $\mathsf{P}_0^{\mathsf{em}}$ from an n-party network Net^n, we provide a communication protocol π^{em}, which implements the emulation functionality $\mathsf{P}_0^{\mathsf{em}}$ from an n-party network Net^n and an intermediate functionality $\mathsf{P}_0^{\mathsf{pm}}$. This functionality $\mathsf{P}_0^{\mathsf{pm}}$ runs the designated party protocol π_0 with a communication protocol π_0^{em} instead of a network Net^{n+1} as described below.

4.4.2.1 Functionality $\mathsf{P}_0^{\mathsf{pm}}$

Functionality $\mathsf{P}_0^{\mathsf{pm}}$ runs (for $t < \frac{n}{2}$, see Fig. 4.4) the emulated party protocol π_0 and a communication protocol π_0^{em}. All interfaces of protocol π_0 (the n I/O-interfaces and the communication interface) are connected to protocol π_0^{em}. The communication protocol π_0^{em} in turn provides n I/O-interfaces corresponding to the emulating parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ which become the n I/O-interfaces of functionality $\mathsf{P}_0^{\mathsf{pm}}$. The concrete communication protocol π_0^{em} we use is described with the protocol π^{em} below.

Adversarial Power		Behavior	
$t < \frac{n}{2}$	CO/IT	$\pi_0 \circ \pi_0^{\mathsf{em}}$	(emulated party protocol)
$\frac{n}{2} \leq t$	CO/IT	$\mathsf{F}^{\mathsf{noSec}}$	(no guarantees)

Figure 4.4: The ideal functionality $\mathsf{P}_0^{\mathsf{pm}}$.

4.4.2.2 Protocol π^{em}

The communication protocol π^{em} (Fig. 4.6) implements a network Net^{n+1} connected to the designated party protocol π_0 (functionality $\mathsf{P}_0^{\mathsf{em}}$, Fig. 4.3) from a communication protocol π_0^{em} connected to the designated party protocol π_0 (functionality $\mathsf{P}_0^{\mathsf{pm}}$, Fig. 4.4) and a network Net^n.

The communication protocol machines π_0^{em} (run by functionality $\mathsf{P}_0^{\mathsf{pm}}$) and π_i^{em} ($i \in [n]$) are designed to interact with each other. Each protocol

machine π_i^{em} connects to the I/O-interface of P_i to $\mathsf{P}_0^{\mathsf{pm}}$ and to the interface of Net^n to P_i. In turn it provides P_i with a communication interface (to the emulated Net^{n+1}) and with an I/O-interface (to the emulated π_0). Recall that $\mathsf{P}_0^{\mathsf{pm}}$ exposes the I/O-interfaces of π_0^{em} as its own I/O-interfaces, as such the π_i^{em} connect directly to the I/O-interfaces of π_0^{em} (for $t < \frac{n}{2}$). Protocol π_0^{em} operates as detailed in Fig. 4.5, the π_i^{em} ($i \in [n]$) are described in Fig. 4.6 below.

Protocol π^{em} emulates Net^{n+1} by making use of the fact that the parties P_i ($i \in [n]$) emulating P_0 are the same parties P_i that are supposed to interact with P_0 via Net^{n+1}. The I/O-interface of P_i to $\mathsf{P}_0^{\mathsf{pm}}$ can therefore serve as a secure channel between P_i (running π_i^{em}) and the emulated P_0 (running π_0^{em}). Protocol π^{em} then integrates P_0 into the network Net^n which is available as a resource by forwarding messages to $\mathsf{P}_0^{\mathsf{pm}}$ (i.e. to π_0^{em}) by means of its I/O-interfaces. Messages, inputs, and outputs between P_i and the emulation of P_0 can directly be forwarded in this fashion. As the emulated party P_0 is only expected to honestly run π_0 for $t < \frac{n}{2}$, the broadcast BC^n available from the resource Net^n can be extended to a broadcast BC^{n+1} by having each π_i^{em} act as a forwarder and performing a majority vote.

Lemma 4.4.1. *Protocol π^{em} UC securely implements $\mathsf{P}_0^{\mathsf{em}}$ from Net^n and $\mathsf{P}_0^{\mathsf{pm}}$ against static adversaries.*

4.4.2.3 Proof of Lemma 4.4.1

In order to prove Lemma 4.4.1, we need to provide a simulator S^{em} such that the ideal model $\mathsf{S}_{\mathcal{A}}^{\mathsf{em}} \circ \mathsf{P}_0^{\mathsf{em}}$ becomes IT indistinguishable from the real model $\pi_{\mathcal{H}}^{\mathsf{em}} \circ \mathsf{Net}^n \circ \mathsf{P}_0^{\mathsf{pm}}$. The simulator S^{em} connects to all interfaces of $\mathsf{P}_0^{\mathsf{em}}$ associated with corrupted parties. The interfaces exposed by $\mathsf{P}_0^{\mathsf{em}}$ are those to the network Net^{n+1} it runs, and, for $t < \frac{n}{2}$, the n I/O-interfaces of the protocol π_0 of P_0. The cases $t \geq \frac{n}{2}$ and $t < \frac{n}{2}$ differ: In case $t \geq \frac{n}{2}$ the simulator S^{em} has to handle the interfaces of corrupted parties among the $\mathsf{P}_1, \ldots, \mathsf{P}_n$ to Net^{n+1} and the P_0 interface to Net^{n+1}. In case $t < \frac{n}{2}$ the simulator S^{em} has to handle the interfaces of corrupted parties to protocol π_0 and to Net^{n+1} (but not the P_0 interface to Net^{n+1}). We can treat both cases jointly if we consider the emulated party P_0 honest for $t < \frac{n}{2}$ and corrupted $t \geq \frac{n}{2}$ as suggested above.

The simulator S^{em} then internally simulates an instance $\widetilde{\mathsf{Net}^n}$ of Net^n and copies $\widetilde{\pi_i^{\mathsf{em}}}$ of π_i^{em} for the honest parties (including P_0 for $t < \frac{n}{2}$).

π_0^{em} connects to *all* interfaces of protocol π_0, that is, to the communication interface and to the n I/O-interfaces corresponding to the parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$. π_0^{em} provides n I/O-interfaces of its own to the n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$. π_0^{em} then processes messages as follows:

Secure Channels: Inputs on the I/O-interface of P_i to π_0^{em} that are labeled as messages from P_i are forwarded to the communication interface of π_0 as messages from P_i. Messages for P_i from the communication interface of π_0 are labeled as message from P_0 and output on the I/O-interface of P_i to π_0^{em}.

Broadcasts: Inputs labeled as broadcast messages from P_i, are forwarded to the communication interface of π_0 as broadcast messages from P_i if received identically at more than $\frac{n}{2}$ I/O-interfaces of π_0^{em} or otherwise ignored. Broadcast messages from the communication interface of π_0 are labeled as broadcast messages from P_0 and output on all I/O-interfaces of π_0^{em}.

I/O for π_0: Unlabeled inputs from the I/O-interface of P_i to π_0^{em} are forwarded to the protocol machine π_0 as input on the I/O-interface of P_i. Outputs from π_0 on the I/O-interface of P_i are output on the I/O-interface of P_i to π_0^{em}.

Figure 4.5: The protocol machine π_0^{em}.

These machines are connected as in protocol π^{em}. Here, for $t < \frac{n}{2}$ the machine $\widetilde{\pi_0^{\mathsf{em}}}$ connects to the other $\widetilde{\pi_i^{\mathsf{em}}}$ like $\mathsf{P}_0^{\mathsf{pm}}$ in the real model and also connects to the I/O-interfaces of corrupted parties to protocol π_0. The simulator S^{em} internally makes use of the communication and the I/O-interface of the $\widetilde{\pi_i^{\mathsf{em}}}$ intended for P_i. The remaining interfaces are exposed to the distinguisher, i.e. the interfaces of corrupted parties to $\widetilde{\mathsf{Net}^n}$, and for $t < \frac{n}{2}$ the interfaces of $\widetilde{\pi_0^{\mathsf{em}}}$ for connecting to the $\widetilde{\pi_i^{\mathsf{em}}}$ of corrupted parties or for $t \geq \frac{n}{2}$ the interfaces of $\widetilde{\pi_i^{\mathsf{em}}}$ for connecting to $\widetilde{\pi_0^{\mathsf{em}}}$.

Now when a (private or BC) message m is received from an honest party P_i via Net^{n+1} the simulator S^{em} inputs m to the communication interface of $\widetilde{\pi_i^{\mathsf{em}}}$ for sending to the appropriate destination. On the other hand, when a $\widetilde{\pi_i^{\mathsf{em}}}$ outputs a (private or BC) message m from a corrupted P_j on the communication interface, this means P_j has sent the message m and simulator S^{em} forwards m to Net^{n+1} via the interface

π_i^{em} connects to the interfaces of Net^n and $\mathsf{P}_0^{\mathsf{pm}}$ belonging to P_i and offers P_i a communication interface to the emulated Net^{n+1} and an I/O-interface to π_0. π_i^{em} then processes messages as follows:

Secure Channels: Messages for P_j arriving on the communication interface are forwarded to P_j via Net^n. Messages for P_0 arriving on the communication interface are labeled as messages from P_i and forwarded to the I/O-interface of $\mathsf{P}_0^{\mathsf{pm}}$. Inputs from the the I/O-interface of $\mathsf{P}_0^{\mathsf{pm}}$ labeled as messages from P_0 are forwarded to the communication interface as messages from P_0. Messages from P_j arriving on the interface to Net^n are forwarded to the communication interface as messages from P_j.

Broadcasts from $\mathsf{P}_1, \ldots, \mathsf{P}_n$: Broadcast messages arriving on the communication interface are forwarded to Net^n as broadcast messages and to the I/O-interface of $\mathsf{P}_0^{\mathsf{pm}}$ labeled as broadcasts from P_i. Broadcast messages from a P_j arriving on the interface to Net^n, unless labeled as originating from P_0, are forwarded both to the communication interface as broadcast messages from P_j and to the I/O-interface of $\mathsf{P}_0^{\mathsf{pm}}$, labeled as broadcast from P_j.

Broadcasts from P_0: Inputs from the the I/O-interface of π_i^{pm} labeled as broadcast messages from P_0 are forwarded as broadcast messages to Net^n, including the label. Broadcast messages arriving on the interface to Net^n labeled as originating from P_0 are forwarded to the communication interface as broadcast message from P_0 if received identically from more than $\frac{n}{2}$ parties P_j, including the copy possibly received from $\mathsf{P}_0^{\mathsf{pm}}$ directly.

I/O for P_0: Inputs from the I/O-interface are forwarded to the I/O-interface of $\mathsf{P}_0^{\mathsf{pm}}$. Unlabeled inputs from the I/O-interface of $\mathsf{P}_0^{\mathsf{pm}}$ are output on the I/O-interface.

Figure 4.6: The protocol machine π_i^{em}.

of P_j. It is fairly straightforward to see that real system and simulation are perfectly indistinguishable.

4.4.3 Implementing Functionality $\mathsf{P}_0^{\mathsf{pm}}$

To complete the emulation of a party P_0, it remains to IT securely implement the functionality $\mathsf{P}_0^{\mathsf{pm}}$ (Fig. 4.4) from a network Net^n by means of a protocol π^{pm}. Recall that functionality $\mathsf{P}_0^{\mathsf{pm}}$ runs the designated party protocol π_0 and the communication protocol π_0^{em} for $t < \frac{n}{2}$, exposing one I/O-interface to each party P_i ($i \in [n]$). For $t \geq \frac{n}{2}$, functionality $\mathsf{P}_0^{\mathsf{pm}}$ turns control over to the adversary.

Any such ideal functionality $\mathsf{P}_0^{\mathsf{pm}}$ can be realized using an MPC protocol π^{pm} that, for $t < \frac{n}{2}$, provides full security in the IT setting. The existence of such a protocol is guaranteed by the following lemma taken from [RB89, Can01]:

Lemma 4.4.2 ([RB89, CDD+99, Can01])**.** *Given a (well-formed [Can01]) ideal functionality* F *there is a protocol* π^{pm} *that implements the ideal functionality* F *with IT full security from a complete and synchronous network of secure channels and a BC channel in the UC setting, against static adversaries corrupting* $t < \frac{n}{2}$ *parties.*

4.5 Implementing a Designated Party MPC

We exhibit a designated party MPC protocol $\pi^{\mathsf{des},\rho}$ which implements an $(n+1)$-party designated party MPC from a common reference string CRS and an $(n+1)$-party network Net^{n+1}. For our purposes, the designated party P_0 (running protocol $\pi_0^{\mathsf{des},\rho}$ as specified below) will be emulated as described in Section 4.4. More formally, we provide a protocol $\pi^{\mathsf{des},\rho}$ that implements the designated party MPC functionality $\mathsf{J}^{\mathsf{des}}$ from a CRS and the emulation functionality $\mathsf{P}_0^{\mathsf{em}}$, running the emulated party protocol $\pi_0 = \pi_0^{\mathsf{des},\rho}$.

4.5.1 Functionality $\mathsf{J}^{\mathsf{des}}$

We define a designated party MPC functionality $\mathsf{J}^{\mathsf{des}}$ that formally captures computing a functionality J with the designated party property.

Functionality $\mathsf{J}^{\mathsf{des}}$ takes as parameter an arbitrary $(n+1)$-party functionality J which has $2n$ interfaces. Here, each party P_i ($i \in [n]$) has one I/O-interface to J and a second I/O-interface formally belonging P_0, but available to P_i. We refer to this second interface as the P_0-interface of P_i

Adversarial Power[21]		Guarantees				
		Cor.	Priv. P_0	Priv. P_i	Fair.	Rob.
IT	$t \leq \rho$	yes[22]	yes	no	yes	yes
IT	$\rho < t < \frac{n}{2}$	yes[22]	yes	no	yes	yes
IT	$\frac{n}{2} \leq t$	no	n/a (corr.)	no	no	no
CO	$t \leq \rho$	yes	yes	yes	yes	yes
CO	$\rho < t < \frac{n}{2}$	yes	yes	yes	yes	no
CO	$\frac{n}{2} \leq t < n - \rho$	yes	n/a (corr.)	yes	no	no
CO	$n - \rho \leq t$	no[23]	n/a (corr.)	no	no	no

Table 4.1: Security Guarantees formalized by functionality $\mathsf{J}^{\mathsf{des}}$.

to J. Functionality $\mathsf{J}^{\mathsf{des}}$ then evaluates functionality J, providing the following guarantees: In case the designated party P_0 is honest, functionality $\mathsf{J}^{\mathsf{des}}$ guarantees privacy of P_0's input, correctness, and fairness against arbitrarily many IT corrupted parties, as well as robustness against $t \leq \rho$ IT corrupted parties. In case the designated party P_0 is corrupted, functionality $\mathsf{J}^{\mathsf{des}}$ still guarantees correctness and privacy to the honest parties against $t < n - \rho$ CO corrupted parties.[21] Recall that, by design of the designated party functionality $\mathsf{P}_0^{\mathsf{em}}$, we think of the emulated party P_0 as honest for $t < \frac{n}{2}$ and corrupted for $t \geq \frac{n}{2}$. Keeping this in mind we arrive at the functionality $\mathsf{J}^{\mathsf{des}}$ described in Fig. 4.7.

Functionality $\mathsf{J}^{\mathsf{des}}$ thus computes the $(n+1)$-party functionality J with properties as summarized in Table 4.1.

4.5.2 Protocol $\pi^{\mathsf{des},\rho}$

We now describe a designated party MPC protocol $\pi^{\mathsf{des},\rho}$ which implements an $(n+1)$-party designated party MPC from a common reference string CRS and an $(n+1)$-party network Net^{n+1}. We then emulate party P_0, having the emulation functionality $\mathsf{P}_0^{\mathsf{em}}$ run protocol ma-

[21] The number t of corrupted parties always pertains to the real parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ and never includes the emulated party P_0.

[22] Correctness is maintained in the sense that the ideal functionality still performs the desired computation. However, the adversary may make inputs dependent on the inputs of honest parties in the current and previous input phases.

[23] Our protocol $\pi^{\mathsf{des},\rho}$ actually achieves correctness here in the sense that it still performs the desired computation. However, the adversary may make inputs dependent on the state of the protocol, i.e. on the inputs of honest parties in previous but not the current input phases. For our subsequent results though, we need not demand correctness here.

Functionality $\mathsf{J}^{\mathsf{des}}$ models computing a functionality J with the designated party property. Like functionality J, functionality $\mathsf{J}^{\mathsf{des}}$ provides one I/O-interface and one P_0-interface to each party P_i ($i \in [n]$). Functionality $\mathsf{J}^{\mathsf{des}}$ operates as follows:[21]

1. If P_0 is corrupted ($t \geq \frac{n}{2}$), and additionally we are in the IT setting or $t \geq n - \rho$, turn control over to the adversary by running functionality $\mathsf{F}^{\mathsf{noSec}}$. Otherwise, run functionality J.
2. Forward any inputs from the I/O-interfaces to J and, in the IT setting, copy them to the adversary.
3. If P_0 is honest ($t < \frac{n}{2}$), forward any inputs from the P_0-interfaces to functionality J, if P_0 is corrupted ($t \geq \frac{n}{2}$), expose all P_0-interfaces of functionality J directly to the adversary, and forward any inputs of honest parties to P_0-interfaces to the adversary.
4. If functionality J makes output:
   (a) If P_0 is honest ($t < \frac{n}{2}$) and $t \leq \rho$ set $o := 1$.
   (b) Elsif P_0 is honest ($\rho < t < \frac{n}{2}$) request an output flag $o \in \{0, 1\}$ (default to $o = 1$) from the adversary.
   (c) Elsif P_0 is corrupted ($\frac{n}{2} \leq t < n - \rho$, in the CO setting) make output to the adversary and take an output flag $o \in \{0, 1\}$ (default to $o = 1$) as input from the adversary.
   (d) If $o = 1$ forward the remaining outputs, otherwise ($o = 0$) halt.
5. Any messages from J to the adversary are forwarded.

Figure 4.7: The ideal functionality $\mathsf{J}^{\mathsf{des}}$.

chine $\pi_0^{\mathsf{des},\rho}$ and provide the network Net^{n+1}. As a result, protocol $\pi^{\mathsf{des},\rho}$ will implement the designated party MPC functionality $\mathsf{J}^{\mathsf{des}}$ from a CRS and emulation functionality $\mathsf{P}_0^{\mathsf{em}}$, running the emulated party protocol $\pi_0 = \pi_0^{\mathsf{des},\rho}$. We obtain protocol $\pi^{\mathsf{des},\rho}$ by adapting the CO MPC protocol of [CLOS02] to our needs ([CLOS02] is in turn an adaption of [GMW87] to the UC setting. For the stand-alone setting we may directly adapt [GMW87] obtaining a stand-alone protocol $\pi_{\mathsf{SA}}^{\mathsf{des},\rho}$.).

4.5.2.1 Modifying [CLOS02]

Before we adapt the protocols of [CLOS02, GMW87] to satisfy the requirements laid out in Section 4.5.1 we first give an overview of [CLOS02, GMW87]:

The protocols in [CLOS02, GMW87] proceed in stages, each consisting of three phases: an input phase, a computation phase, and an output phase. If the functionality to be implemented is non-interactive, a single stage suffices; interactive functionalities require several stages. In the input phase the players commit to their inputs and share them among the participants according to a prescribed secret sharing scheme. In [CLOS02] this is a simple XOR n-out-of-n sharing, but as described in [Gol04] a different sharing can be used to trade privacy for robustness. In the computation phase, [CLOS02, GMW87] use oblivious transfer (OT) to evaluate the desired function on the inputs. All intermediate results are computed as sharings, where the parties commit to their share and prove it correct using zero-knowledge (ZK) proofs thus achieving security against active adversaries. In the output phase the results of the computation are reconstructed by the players by opening their commitments to the shares of the final result.

The security requirements of Section 4.5.1 for protocol $\pi^{\mathsf{des},\rho}$ can be grouped into four cases:

1. In the CO case where P_0 is honest we require full security for $t \leq \rho$ and tolerate only the loss of robustness beyond that bound.
2. In the CO case where P_0 is corrupted, we only require privacy and correctness up to $t < n - \rho$.
3. In the IT case where P_0 is honest, we require correctness, privacy for P_0, fairness, and robustness for up to $t \leq \rho$. Again, we tolerate the loss of robustness beyond that bound.
4. In the IT case where P_0 is corrupted, we do not require any security guarantees.

We show that the MPC protocols of [CLOS02, GMW87] can be modified accordingly. Like [GMW87] the MPC protocol in [CLOS02] operates on shares, utilizes oblivious transfer (OT) for multiplications, and uses the compiler of [GMW87] which is based on commitments and zero-knowledge (ZK) proofs to achieve security against active adversaries. We

demonstrate how these components can be modified to provide additional guarantees without compromising their original security properties.

4.5.2.2 Modifying the Computational Primitives

We need to modify [CLOS02, GMW87] such that privacy and correctness are IT for player P_0. All three CO primitives employed in [CLOS02, GMW87] (i.e. OT, commitments, and ZK proofs) can be implemented CO securely while IT protecting one (in our case always P_0) of the participants. That is, we can implement [CLOS02, GMW87] using primitives that remain secure if P_0 is involved in their computation and honest, even if arbitrarily many other players P_i are IT corrupted.

This serves our purpose: Using such primitives is merely a refinement of [CLOS02, GMW87], thus the resulting $(n+1)$-party protocol is still correct and private in presence of arbitrarily many actively corrupted parties in the CO setting. Furthermore, given these modifications, the protocol is private for player P_0 even in presence of arbitrarily many IT corrupted parties. Finally, as long as player P_0 is honest, the protocol is correct in the IT setting.

Below we discuss suitable CO primitives for [CLOS02, GMW87], namely OT, commitments, (perfectly) zero-knowledge arguments of knowledge (ZK-AoK), and CO zero-knowledge proofs of knowledge (cZK-PoK) which IT protect the designated party P_0.

Oblivious Transfer. As shown in [CLOS02, Section 4.1.1] the OT protocol of [GMW87, Gol04, pp. 640–643] is UC secure. Furthermore, it is easy to see that it IT protects the receiver. The [CLOS02, GMW87] protocols make no restriction as to which participant of an OT execution acts as sender or receiver. So we may use said OT protocol and still IT protect P_0 by making P_0 the receiver in every invocation of OT involving P_0. Alternatively, a UC secure OT protocol that IT protects the sender can be obtained by "turning around" the above OT as shown in [Wul07, Theorem 4.1]. Thus, security for P_0 is guaranteed in any OT invocation, even with an IT adversary.

Commitment. For [CLOS02], we use the UC secure, one-to-many IT hiding and IT binding commitment schemes in the CRS-model described

in Section A.1. For our purpose, we employ the IT binding variation for commitments issued by the parties P_i ($i \in [n]$) and the IT hiding variation for commitments issued by the designated party P_0. Thus we obtain a UC secure realization of the one-to-many commitment functionality $\mathsf{F}_{\mathsf{Com},1:\mathsf{M}}$ that guarantees security for P_0 against any IT adversary.

In the stand-alone case, for [GMW87], we may directly use IT hiding and IT binding commitment schemes which do not rely on a CRS.

ZK proofs. [CLOS02, Prop. 9.4] shows how to UC securely implement the ZK functionality $\mathsf{F}_{\mathsf{ZK},1:\mathsf{M}}$ from the commitment functionality $\mathsf{F}_{\mathsf{Com},1:\mathsf{M}}$ without use of further CO assumptions.[24] The one-to-many ZK protocol of [CLOS02] is based on the two party protocol of [CF01, Section 5]. This protocol in turn is based on the Hamiltonian Cycles ZK proof. Hence, on the one hand, when using an IT binding commitment scheme, we obtain cZK-PoKs. On the other hand, when using an IT hiding commitment scheme, we obtain ZK-AoKs. The one-to-many property is obtained by repeating the two-party protocol with each player over the broadcast channel. In addition, the proof is accepted only if the transcripts of all invocations constitute valid proofs. Now, if P_0 is the prover, we instantiate the one-to-many commitment functionality $\mathsf{F}_{\mathsf{Com},1:\mathsf{M}}$ with the IT hiding scheme described above. Otherwise, we instantiate $\mathsf{F}_{\mathsf{Com},1:\mathsf{M}}$ with the IT binding scheme. Thus we obtain a protocol that is always secure for P_0, even against any IT adversary.

In the stand-alone case, for [GMW87], we may directly use cZK-PoKs and ZK-AoKs which do not rely on a CRS.

Commit-and-Prove. Instead of directly working with commitment and ZK functionalities (as [GMW87] does), [CLOS02] introduces a new primitive called one-to-many commit-and-prove $\mathsf{F}_{\mathsf{CP},1:\mathsf{M}}$. [CLOS02, Section 7.1] provides a protocol implementing the two-party[25] functionality F_{CP} secure against static adversaries, which relies only on F_{ZK} and a standard commitment scheme (together with the corresponding computational assumptions). Since this protocol is non-interactive, it can easily be extended into a one-to-many protocol by having the sender broadcast all

[24] Actually, this protocol is secure against adaptive adversaries. For static adversaries a non-interactive protocol might be used. However, for ease of discussion, we directly use the adaptive protocol.

[25] The one-to-many protocol presented in [CLOS02, Prop. 9.5] encompasses adaptive adversaries.

messages and use $\mathsf{F}_{\mathsf{ZK},1:\mathsf{M}}$ instead of F_{ZK}. Hence, when implementing $\mathsf{F}_{\mathsf{ZK},1:\mathsf{M}}$ as described above, we can use IT hiding or IT binding commitments in the implementation of $\mathsf{F}_{\mathsf{CP},1:\mathsf{M}}$ to IT protect the designated party P_0. Further CO assumptions are not required. Thus, we can implement the commit-and-prove functionality $\mathsf{F}_{\mathsf{CP},1:\mathsf{M}}$ UC securely for P_0, even against any IT adversary.

4.5.2.3 Modifying Sharing and Output Reconstruction

We now describe how to modify the sharing and output reconstruction underlying [CLOS02, GMW87] in order to meet our robustness and fairness requirements.

We have to robustly tolerate up to $t \leq \rho$ corruptions among the parties P_i ($i \in [n]$), while preserving the unconditional privacy of P_0. This can be accomplished by modifying the underlying sharing of [CLOS02, GMW87] as described in [Gol04, Section 7.5.5]. We use a sharing where any set M of $n - \rho + 1$ parties that *includes* P_0 is qualified, i.e. can reconstruct. Such a sharing can efficiently be implemented using a $(2n-\rho)$-out-of-$(2n)$ Shamir-sharing where P_0 receives n shares and each remaining party P_i obtains a single share. Here, we inherently trade privacy for robustness: Any qualified set M of parties can reconstruct the input of the remaining parties. So any qualified set M of honest parties can recover the input of up to ρ corrupted parties P_i ($i \in [n]$). This ensures robustness, should up to $t \leq \rho$ corrupted parties try to disrupt the computation. On the other hand, any such qualified set M of corrupted parties can violate the privacy of the remaining parties.

Finally we have to guarantee fairness whenever P_0 is honest. As noted in [Gol04] only the player opening his commitments last in the output phase can violate fairness. If we specify that P_0 should open last, and only if he can contribute sufficiently many shares that all players can reconstruct the result, then the resulting protocol $\pi^{\mathsf{des},\rho}$ is fair in the IT setting as long as P_0 is honest.

4.5.3 The Security of the Designated Party Protocol $\pi^{\mathsf{des},\rho}$

In summary, we have constructed a protocol $\pi^{\mathsf{des},\rho}$ from [CLOS02] securely implementing $\mathsf{J}^{\mathsf{des}}$ in the UC setting:

Lemma 4.5.1. *For any robustness parameter $\rho < \frac{n}{2}$ there is a protocol $\pi^{\mathsf{des},\rho}$ that implements $\mathsf{J}^{\mathsf{des}}$ UC securely against static adversaries from a* CRS *setup and an emulation functionality $\mathsf{P}_0^{\mathsf{em}}$ running $\pi_0^{\mathsf{des},\rho}$ as designated party protocol.*

Furthermore, we have constructed a protocol $\pi_{\mathsf{SA}}^{\mathsf{des},\rho}$ from [GMW87] securely implementing $\mathsf{J}^{\mathsf{des}}$ in the stand-alone setting, *without* reliance on a CRS:

Lemma 4.5.2. *For any robustness parameter $\rho < \frac{n}{2}$ there is a protocol $\pi_{\mathsf{SA}}^{\mathsf{des},\rho}$ that implements $\mathsf{J}^{\mathsf{des}}$ stand-alone securely against static adversaries from a designated party functionality $\mathsf{P}_0^{\mathsf{em}}$ running $\pi_{\mathsf{SA},0}^{\mathsf{des},\rho}$ as designated party protocol.*

A proof-sketch of the above lemmata can be found below. CO assumptions sufficient for implementing the necessary CO primitives for protocol $\pi^{\mathsf{des},\rho}$, in particular perfectly hiding or perfectly binding UC secure commitments [DN02], are for instance the p-subgroup assumption or the decisional composite residuosity assumption. For the stand-alone setting, weaker assumptions, e.g. enhanced trapdoor one-way permutations are sufficient [Gol04]. A similar approach, where *all* players use primitives that IT disclose no undesired information is used in [KMQR09] to achieve long-term security for specific functions.

4.5.4 Proof Sketch for Lemmata 4.5.1 and 4.5.2

We will now show that the modified versions of [CLOS02] and [GMW87] described in Section 4.5.2.1 fulfill the requirements stated in Lemmata 4.5.1 and 4.5.2 respectively.

Case 1: CO security with robustness for $t \leq \rho$ and fairness beyond, P_0 honest ($t < \frac{n}{2}$). This claim follows immediately from the IT security guarantees of protocol $\pi^{\mathsf{des},\rho}$ shown in Case 3, and the CO security guarantees shown in Case 2.

Case 2: CO security with abort for $t < n - \rho$ corrupted parties, P_0 corrupted ($t \geq \frac{n}{2}$). CO correctness and privacy are already implied by [CLOS02, GMW87]. Our modifications to CO primitives and opening procedures are within the limits of the original protocol and only apply restrictions as to what kinds of primitives are used in specific situations. The modification to the sharing can be treated as in [Gol04]. As shares

observed by corrupted parties are still uniformly random for $t < n - \rho$ corrupted parties the modifications to the simulator remain trivial. As such the proof in [CLOS02, GMW87] remains applicable with minimal modifications and we obtain a CO secure implementation of the ideal functionality in this case.

Note that agreement on abort is achieved: The only way to make a party abort is to send an incorrect message (one for which the zero-knowledge proof does not hold). However, since the message together with the proof is sent over a BC channel, this will be noted by all honest parties and they will all abort.

Case 3: IT security for honest P_0 ($t < \frac{n}{2}$) with robustness for $t \leq \rho$, and with fairness for $t < n - \rho$. We sketch a simulator to demonstrate that [CLOS02], tweaked as described above, UC securely implements the ideal functionality $\mathsf{J}^{\mathsf{des}}$ in the given setting, i.e. an IT adversary corrupting $t \leq n-\rho$ parties not including party P_0. We have to show that correctness, privacy for P_0, and fairness are guaranteed. In case of $t \leq \rho$ we have to show that robustness is maintained as well. The proof for [GMW87] in the stand-alone setting works analogously, but relies on rewinding to facilitate simulation, instead of extractability and equivocability of UC commitments by means of the CRS. Hence we refrain from describing a separate simulator for the stand-alone setting.

The simulator will receive the inputs of all honest parties except P_0 from the ideal functionality $\mathsf{J}^{\mathsf{des}}$. Furthermore corrupted parties have to commit to their input using binding UC commitments[26], so by extractability the simulator can extract their inputs and forward them to $\mathsf{J}^{\mathsf{des}}$. The simulator then simulates the protocol machines of all honest parties, with an *arbitrary* input x'_0 for the simulated party P_0, and the inputs of the other honest parties as obtained from $\mathsf{J}^{\mathsf{des}}$. The simulation then proceeds up to the point where an output y is opened, i.e. where the simulator receives a y from $\mathsf{J}^{\mathsf{des}}$. Recall that party P_0, which is honest by assumption and thus simulated internally by the simulator, is supposed to broadcast its opening information last, and only if sufficiently many (i.e. $n - \rho$) parties have broadcasted their opening information correctly, in order to guarantee correct reconstruction of y.

We first consider the case where $\rho < t \leq n - \rho$ parties are corrupted. Thus, we only need to guarantee fairness. If at least $t-\rho$ corrupted parties

[26] Providing extractability and equivocability by means of the CRS.

broadcast their opening information correctly, then the simulator makes use of the equivocability of commitments to have the internally simulated P_0 open to the output y and sets the output flag to $o = 1$, otherwise it sets $o = 0$.

Finally, given that $t \leq \rho$ parties are corrupted, the adversary can no longer prevent the honest parties from reconstructing the output. Hence, P_0, regardless of the shares distributed by the adversary, opens the output to y, again making use of the equivocability of commitments.

The behavior of the simulator is IT indistinguishable from the real protocol. As in the real protocol, the designated party P_0 in the simulation only converses with the other parties by means of hiding commitments, ZK proofs and OT invocations IT protecting party P_0. Furthermore, the sharing scheme is such that without cooperation of the designated party P_0, which is honest by assumption, no information can be recovered. As such no information whatsoever is disseminated by the designated party P_0 to the corrupted parties until reconstruction takes place in an opening phase.

Case 4: IT, P_0 corrupted ($t \geq \frac{n}{2}$), no security guarantees. As we make no security guarantees in this case, there is nothing to show.

4.6 Implementing a Hybrid-Secure MPC

It remains to provide a protocol π_i^{in} implementing a hybrid-secure MPC functionality F^{ρ} (Fig. 4.1) from the designated party MPC functionality $\mathsf{J}^{\mathsf{des}}$. Protocol π^{in} does so by ensuring IT privacy for $t < \frac{n}{2}$ and CO privacy for $t < n - \rho$. This is achieved by having π_i^{in} share any input x_i as $x_i = x_i^{\mathsf{em}} \oplus x_i^{\mathsf{des}}$, where x_i^{em} is chosen uniformly at random over the input space.[27] Protocol π_i^{in} then inputs x_i^{des} at the I/O-interface of functionality $\mathsf{J}^{\mathsf{des}}$, while entering the share x_i^{em} via the P_0-interface of functionality $\mathsf{J}^{\mathsf{des}}$. As functionality $\mathsf{J}^{\mathsf{des}}$ guarantees IT privacy for P_0, this results in a protocol where privacy is IT as long as P_0 is honest, i.e. for $t < \frac{n}{2}$. At the same time the CO privacy of all parties is guaranteed by functionality $\mathsf{J}^{\mathsf{des}}$ for $t < n - \rho$. Hence, we obtain a protocol with CO privacy for $t < n - \rho$.

[27] Wlog, we assume a group structure with operation $\oplus$ over the input space. Assuming inputs from a finite field is a common convention in MPC, or we may think of bitstrings, with XOR as operation.

To maintain CO correctness for $t \geq \frac{n}{2}$ (when the emulated party P_0 is corrupted) additional measures are needed: For $t \geq \frac{n}{2}$, functionality $\mathsf{J}^{\mathsf{des}}$ turns the P_0-interface over to the adversary, who could manipulate the x_i^{em} at will, effectively manipulating the inputs x_i to produce *incorrect* results. We solve this problem by using commitments. So, in the following let commit and open denote the respective procedures for a UC secure IT hiding commitment scheme (see [DN02], App. A.1). Then, π_i^{in} may compute an IT hiding commitment $(c_i, o_i) = \mathsf{commit}(x_i^{\mathsf{em}})$ to x_i^{em}. Protocol π_i^{in} inputs the commitment c_i together with x_i^{em} at the P_0-interface of functionality $\mathsf{J}^{\mathsf{des}}$ while entering the matching opening information o_i together with x_i^{des} at the I/O-interface of functionality $\mathsf{J}^{\mathsf{des}}$. We then have functionality $\mathsf{J}^{\mathsf{des}}$ check these commitments. In case a commitment fails to open correctly, we can abort the computation. This construction achieves CO correctness because a CO adversary controlling the P_0-interfaces cannot open such a commitment incorrectly. At the same time, the unconditional privacy of the x_i^{em} is unaffected as the commitments c_i are IT hiding.

Finally, we need to guarantee robustness for $t \leq \rho$. Thus, we may not abort if a commitment c_i fails to open correctly. Instead, the functionality $\mathsf{J}^{\mathsf{des}}$ outputs a complaint, requesting that P_i directly inputs x_i via the I/O-interface of $\mathsf{J}^{\mathsf{des}}$. This procedure does not affect privacy since commitments c_i only fail to open correctly if either P_i is corrupted or if the emulated party P_0 is controlled by the adversary. In the first case, we need not guarantee privacy to P_i. In the latter case, we have $t \geq \frac{n}{2}$, so we only need to guarantee CO privacy, which $\mathsf{J}^{\mathsf{des}}$ already does. Correctness is maintained since privacy is maintained and a party can only replace its own input.

The fairness properties of $\mathsf{J}^{\mathsf{des}}$ are unaffected by the measures described above, so the resulting protocol is fair whenever the emulated party P_0 is honest, i.e. whenever $t < \frac{n}{2}$.

Summarizing the measures above, we obtain an input protocol π^{in} (Fig. 4.8) and a matching functionality J to be run by functionality $\mathsf{J}^{\mathsf{des}}$. Protocol π^{in} takes care of sharing inputs, providing commitments and answering complaints. Functionality J reconstructs the inputs, checks commitments, makes complaints, and finally evaluates the target functionality F.

Lemma 4.6.1. *For any robustness parameter $\rho < \frac{n}{2}$ protocol π^{in} UC securely implements F^{ρ} against static adversaries from a designated party MPC functionality $\mathsf{J}^{\mathsf{des}}$ running functionality J.*

Protocol machine π_i^{in} connects to the I/O- and P_0-interfaces of P_i to functionality $\mathsf{J}^{\mathsf{des}}$. In turn π_i^{in} offers an I/O-interface to P_i. Protocol machine π_i^{in} then proceeds as follows:

1. On receiving an input on the I/O-interface: Choose x_i^{em} uniformly at random and compute $x_i^{\mathsf{des}} := x_i \oplus x_i^{\mathsf{em}}$. Using an IT hiding commitment scheme compute $[c_i, o_i] = \mathsf{commit}(x_i^{\mathsf{em}})$. Pass input (x_i^{em}, o_i) to the P_0-interface and $(x_i^{\mathsf{des}}, c_i)$ to the I/O-interface of $\mathsf{J}^{\mathsf{des}}$. Receive a complaint vector $\vec{e}$ on the I/O-interface of $\mathsf{J}^{\mathsf{des}}$. If $e_i = 0$ then input x_i to the I/O-interface of $\mathsf{J}^{\mathsf{des}}$.
2. On receiving an output on the I/O-interface of $\mathsf{J}^{\mathsf{des}}$, forward y to the I/O-interface to P_i.

Figure 4.8: The protocol machine π_i^{in}.

Functionality J connects to the n I/O-interfaces of functionality F and in turn provides one P_0-interface and one I/O-interface per party P_i. Functionality J then proceeds as follows:

1. Run functionality F.
2. On receiving input on an I/O-interface in a given round: Parse inputs on the I/O-interfaces of the P_i as $(x_i^{\mathsf{des}}, c_i)$. Parse inputs on the P_0-interfaces of the P_i as (x_i^{em}, o_i). Output a complaint vector $\vec{e} = (x_i^{\mathsf{em}} \stackrel{?}{=} \mathsf{open}(c_i, o_i))_{i \in [n]}$ via the I/O-interfaces of the P_i. For all $i \in [n]$ where $e_i = 1$ compute $x_i := x_i^{\mathsf{des}} \oplus x_i^{\mathsf{em}}$. Take new inputs x_i on the I/O-interfaces of P_i where $e_i = 0$, default to $x_i = \bot$ if no input is provided. Forward all inputs $x_i \neq \bot$ to functionality F.
3. On receiving an output y from F, forward y to the I/O-interfaces of the P_i.
4. Forward any messages from F to the adversary.

Figure 4.9: The functionality J.

4.6.1 Proof of Lemma 4.6.1

We have to show that the protocol π^{in} implements F^{ρ} from functionality $\mathsf{J}^{\mathsf{des}}$ for the choice of J described in Fig. 4.9. We do so by providing

an appropriate simulator S^{in} that renders the ideal model $\mathsf{S}^{\mathsf{in}} \circ \mathsf{F}^{\rho}$ indistinguishable from the real model $\pi^{\mathsf{in}}_{\mathcal{H}} \circ \mathsf{J}^{\mathsf{des}}$. We treat the settings $t < \frac{n}{2}$ and $\frac{n}{2} \leq t < n - \rho$ separately. For $t \geq n - \rho$ functionality F^{ρ} gives up, so there is nothing to show. For $\frac{n}{2} \leq t < n - \rho$, we have to show that protocol π^{in} implements F^{ρ} in the CO setting. For $t < \frac{n}{2}$, it suffices to show that protocol π^{in} implements F^{ρ} in the IT setting.

4.6.1.1 Proof of Lemma 4.6.1 for $\frac{n}{2} \leq t < n - \rho$

We show that, for $\frac{n}{2} \leq t < n - \rho$, there is a simulator S^{in} which renders ideal model $\mathsf{S}^{\mathsf{in}} \circ \mathsf{F}^{\rho}$ indistinguishable from the real model $\pi^{\mathsf{in}}_{\mathcal{H}} \circ \mathsf{J}^{\mathsf{des}}$ in the CO setting.

In the CO setting, for $\frac{n}{2} \leq t < n - \rho$, functionality $\mathsf{J}^{\mathsf{des}}$ is correct and private for inputs at its I/O-interfaces, but gives the adversary control over inputs at its P_0-interfaces (we may consider the emulated party P_0 corrupted) and guarantees no robustness or fairness, only agreement on abort.

The simulator S^{in} is connected to the interfaces of the corrupted parties to the ideal functionality F^{ρ}. In turn the simulator S^{in} simulates the I/O-interfaces of functionality $\mathsf{J}^{\mathsf{des}}$ belonging to corrupted parties and the P_0-interface of functionality $\mathsf{J}^{\mathsf{des}}$ to the distinguisher.

For $\frac{n}{2} \leq t < n - \rho$, the simulator S^{in} then operates as follows:

1. When an honest party makes input to F^{ρ} or the distinguisher makes input via the I/O-interface of a corrupted party:
   (a) For all honest parties P_i making input, choose $\tilde{x}_i^{\mathsf{em}}$ at random and compute IT hiding commitments $(c_i, \tilde{o}_i) = \mathsf{commit}(\tilde{x}_i^{\mathsf{em}})$.
   (b) Give the $(\tilde{x}_i^{\mathsf{em}}, \tilde{o}_i)$ as output to the distinguisher over the P_0-interface.
   (c) Receive some (x_i^{em}, o_i) from the distinguisher over the P_0-interface.
   (d) Receive some $(x_i^{\mathsf{des}}, c_i)$ from the distinguisher over the I/O-interfaces of the $\mathsf{P}_i \in \mathcal{A}$.
   (e) Output a complaint vector $\vec{e} = (x_i^{\mathsf{em}} \stackrel{?}{=} \mathsf{open}(c_i, o_i))_{i \in [n]}$ to the distinguisher via the I/O-interfaces.
   (f) Receive an output flag o from the distinguisher, default to $o = 1$ if none is provided. In case $o = 0$, forward o to F^{ρ} and halt.

(g) For the $\mathsf{P}_i \in \mathcal{A}$ where $e_i = 1$ compute $x_i := x_i^{\mathsf{des}} \oplus x_i^{\mathsf{em}}$.

(h) Take new inputs x_i on the I/O-interfaces of the $\mathsf{P}_i \in \mathcal{A}$ where $e_i = 0$, default to $x_i = \perp$ if no input is provided.

(i) Forward all inputs $x_i \neq \perp$ ($i \in \mathcal{A}$) to functionality F.

2. When functionality F^ρ makes output:

(a) Forward the output y of F^ρ to the distinguisher via the I/O-interfaces of the $\mathsf{P}_i \in \mathcal{A}$

(b) Receive an output flag o from the distinguisher, default to $o = 1$ if no output flag is provided.

(c) Forward the output flag o to F^ρ, and in case $o = 0$ halt.

We now argue that the simulator S^{in} indeed renders the ideal model $\mathsf{S}^{\mathsf{in}} \circ \mathsf{F}^\rho$ indistinguishable from the real model $\pi_{\mathcal{H}}^{\mathsf{in}} \circ \mathsf{J}^{\mathsf{des}}$.

When input is made by some party P_i, protocol machine π_i^{in} in $\pi_{\mathcal{H}}^{\mathsf{in}} \circ \mathsf{J}^{\mathsf{des}}$ first splits its input into $x_i = x_i^{\mathsf{des}} \oplus x_i^{\mathsf{em}}$ (where x_i^{em} is uniformly random) and computes the IT hiding commitment $(c_i, o_i) = \mathsf{commit}(x_i^{\mathsf{em}})$. Then, π_i^{in} provides (x_i^{em}, o_i) as input to functionality $\mathsf{J}^{\mathsf{des}}$ at the P_0-interface which is controlled by the adversary. S^{in} simulates this indistinguishably by providing random values $(\tilde{x}_i^{\mathsf{em}}, \tilde{o}_i)$ with appropriate opening information to the distinguisher over the P_0-interface.

Furthermore, protocol machine π_i^{in} provides $(x_i^{\mathsf{des}}, c_i)$ as input to $\mathsf{J}^{\mathsf{des}}$ via the I/O-interface. Functionality $\mathsf{J}^{\mathsf{des}}$ then issues a boolean complaint vector $\vec{e} = (x_i^{\mathsf{em}} = \mathsf{open}(c_i, o_i))_{i \in [n]}$, indicating for which parties the opening failed. The complaint vector $\vec{e}$ is first handed to the adversary and upon receipt of an output flag $o = 1$ to the remaining parties. Functionality $\mathsf{J}^{\mathsf{des}}$ then allows these parties to answer the complaint with a new x_i, and computes $x_i = x_i^{\mathsf{des}} \oplus x_i^{\mathsf{em}}$ for the remaining parties. S^{in} simulates this behavior identically to the corrupted parties. Finally the ideal functionality $\mathsf{J}^{\mathsf{des}}$ forwards the x_i to F, which simulator S^{in} simulates by inputting the x_i to F^ρ.

This simulation is faithful as long as the adversary does not manage to open a commitment c_i to a value other than x_i^{em} (which being CO bounded it cannot).[28]

[28] The commitments to the x_i^{em} and the complaint procedure guarantee that the computation is carried out with correct values x_i^{em}. That is, the input shares x_i^{des} and x_i^{em} have the relation $x_i^{\mathsf{des}} \oplus x_i^{\mathsf{em}} = x_i$. Otherwise, if the adversary controls the P_0-interfaces (as is the case here), he could manipulate the values x_i^{em} leading to a computation with wrong inputs x_i and hence to an incorrect result.

When output is made, functionality $\mathsf{J}^{\mathsf{des}}$ delivers the output y to the adversary and awaits an output flag deciding output delivery to honest parties. Outputs are simply forwarded by π_i^{in}. Functionality F^{ρ} behaves identically and as such the simulator S^{in} need only forward the messages in question.

Hence the protocol π^{in} CO securely implements the functionality F^{ρ} for $\frac{n}{2} \leq t \leq n - \rho$.

4.6.1.2 Proof of Lemma 4.6.1 for $t < \frac{n}{2}$

We show that, for $t < \frac{n}{2}$, there is a simulator S^{in} which renders ideal model $\mathsf{S}^{\mathsf{in}} \circ \mathsf{F}^{\rho}$ indistinguishable from the real model $\pi_{\mathcal{H}}^{\mathsf{in}} \circ \mathsf{J}^{\mathsf{des}}$ in the IT setting.

In the IT setting for $t < \frac{n}{2}$, functionality $\mathsf{J}^{\mathsf{des}}$ is fair, correct and private for inputs at its P_0-interfaces (we may consider the emulated party P_0 honest) but forwards inputs at its I/O-interfaces to the adversary. For $t \leq \rho$, functionality $\mathsf{J}^{\mathsf{des}}$ is additionally robust.

The simulator S^{in} is connected to the interfaces of the corrupted parties to the ideal functionality F^{ρ}. In turn the simulator S^{in} simulates the I/O- and P_0-interfaces of functionality $\mathsf{J}^{\mathsf{des}}$ belonging to corrupted parties to the distinguisher.

For $t < \frac{n}{2}$, the simulator S^{in} then operates as follows:

1. When an honest party makes input to F^{ρ} or the distinguisher makes input via the I/O-interface of a corrupted party:
   (a) For all honest parties P_i making input, choose x_i^{des} and x_i^{em} at random and compute IT hiding commitments $(c_i, o_i) = \mathsf{commit}(x_i^{\mathsf{em}})$.
   (b) Give the $(x_i^{\mathsf{des}}, c_i)$ as output to the distinguisher.
   (c) Receive inputs (x_i^{em}, o_i) and $(x_i^{\mathsf{des}}, c_i)$ from the distinguisher over the P_0- and I/O-interfaces of the corrupted parties $\mathsf{P}_i \in \mathcal{A}$ respectively.
   (d) For $\rho < t < \frac{n}{2}$, request an output flag o from the distinguisher, default to $o = 1$ if none is provided. In case $o = 0$, forward o to F^{ρ} and halt.
   (e) Output a complaint vector $\vec{e} = (x_i^{\mathsf{em}} \stackrel{?}{=} \mathsf{open}(c_i, o_i))_{i \in [n]}$ to the distinguisher via the I/O-interfaces.

(f) For the $\mathsf{P}_i \in \mathcal{A}$ where $e_i = 1$ compute $x_i := x_i^{\mathsf{des}} \oplus x_i^{\mathsf{em}}$.

(g) Take new inputs x_i on the I/O-interfaces of the $\mathsf{P}_i \in \mathcal{A}$ where $e_i = 0$, default to $x_i = \perp$ if no input is provided.

(h) Forward all inputs $x_i \neq \perp$ ($i \in \mathcal{A}$) to functionality F.

2. When functionality F^ρ makes output

(a) For $\rho < t < \frac{n}{2}$, request an output flag o from the distinguisher, default to $o = 1$ if none is provided. Forward o to F^ρ and, in case $o = 0$, halt.

(b) Forward the output y of F^ρ to the distinguisher via the I/O-interfaces of the $\mathsf{P}_i \in \mathcal{A}$.

We now argue that the simulator S^{in} indeed renders the ideal model $\mathsf{S}^{\mathsf{in}} \circ \mathsf{F}^\rho$ indistinguishable from the real model $\pi_{\mathcal{H}}^{\mathsf{in}} \circ \mathsf{J}^{\mathsf{des}}$.

When input is made by some party P_i, protocol machine π_i^{in} in $\pi_{\mathcal{H}}^{\mathsf{in}} \circ \mathsf{J}^{\mathsf{des}}$ first splits its input into $x_i = x_i^{\mathsf{des}} \oplus x_i^{\mathsf{em}}$ (where x_i^{em} is uniformly random) and computes the IT hiding commitment $(c_i, o_i) = \mathsf{commit}(x_i^{\mathsf{em}})$. Then, π_i^{in} provides (x_i^{em}, o_i) and $(x_i^{\mathsf{des}}, c_i)$ as input to $\mathsf{J}^{\mathsf{des}}$ via the P_0- and the I/O-interface of P_i to $\mathsf{J}^{\mathsf{des}}$ respectively. In the current context, for $t < \frac{n}{2}$ in the IT setting, $\mathsf{J}^{\mathsf{des}}$ forwards $(x_i^{\mathsf{des}}, c_i)$ to the adversary. S^{in} simulates this indistinguishably by providing random values $(x_i^{\mathsf{des}}, c_i)$ to the distinguisher. Here it is important to note that c_i is a hiding commitment, and as such really independent of x_i^{em}.

Functionality $\mathsf{J}^{\mathsf{des}}$ requests an output flag o and for $o = 1$ issues a boolean complaint vector $\vec{e} = (x_i^{\mathsf{em}} = \mathsf{open}(c_i, o_i))_{i \in [n]}$, indicating for which parties the opening failed. Functionality $\mathsf{J}^{\mathsf{des}}$ then allows these parties to answer the complaint with a new x_i, and computes $x_i = x_i^{\mathsf{des}} \oplus x_i^{\mathsf{em}}$ for the remaining parties. Note that for $t < \frac{n}{2}$ no honest party will ever receive a complaint when trying to give input. S^{in} simulates this behavior identically to the corrupted parties. Finally the ideal functionality $\mathsf{J}^{\mathsf{des}}$ forwards the x_i to F, which simulator S^{in} simulates by inputting the x_i to F^ρ.

When output is made, functionality $\mathsf{J}^{\mathsf{des}}$ for $\rho < t < \frac{n}{2}$ requests an output flag o from the distinguisher, defaulting to $o = 1$ if none is provided. In case $o = 0$, functionality $\mathsf{J}^{\mathsf{des}}$ halts. Otherwise $\mathsf{J}^{\mathsf{des}}$ delivers the output y to all parties, Outputs are simply forwarded by π_i^{in}. Functionality F^ρ behaves identically, so the simulator S^{in} need only forward the messages in question.

Hence the protocol π^{in} IT securely implements the functionality F^ρ for $t < \frac{n}{2}$.

4.7 The Security of the Stand-Alone Protocol π^{ρ}_{SA}

Having shown the UC security of protocol π^{ρ} as claimed in Theorem 4.3.1, we now prove the SA security of protocol π^{ρ}_{SA} as claimed in Theorem 4.3.2. Thus, we have to show that the protocol $\pi^{\rho}_{\mathsf{SA}} = \pi^{\mathsf{in}} \circ \pi^{\mathsf{des},\rho}_{\mathsf{SA}} \circ \pi^{\mathsf{em}} \circ \pi^{\mathsf{pm}}$ stand-alone securely implements the hybrid MPC functionality F^{ρ} from an n-party network Net^{n}.

The SA secure protocol π^{ρ}_{SA} differs from the UC secure protocol π^{ρ} only with respect to the subprotocol $\pi^{\mathsf{des},\rho}$. In π^{ρ}_{SA} we use a variation $\pi^{\mathsf{des},\rho}_{\mathsf{SA}}$ of subprotocol $\pi^{\mathsf{des},\rho}$, which we construct from the protocol of [GMW87] instead of [CLOS02] as described in Section 4.5. As protocol $\pi^{\mathsf{des},\rho}$ is the only subprotocol of protocol π^{ρ} relying on the CRS we obtain a hybrid MPC protocol π^{ρ}_{SA} which is only SA secure, but does not rely on a CRS.

Because the emulation of a party as implemented by the subprotocols π^{pm} and π^{em} remains unmodified, it is sufficient to prove that the subprotocols π^{in} and $\pi^{\mathsf{des},\rho}_{\mathsf{SA}}$ stand-alone securely implement the hybrid MPC functionality F^{ρ} from an emulated party functionality $\mathsf{P}^{\mathsf{em}}_{0}$ running the emulated party protocol $\pi_0 = \pi^{\mathsf{des},\rho}_{\mathsf{SA},0}$.

We have shown in Section 4.5 that protocol $\pi^{\mathsf{des},\rho}_{\mathsf{SA}}$ stand-alone securely implements functionality $\mathsf{J}^{\mathsf{des}}$ from an emulated party functionality $\mathsf{P}^{\mathsf{em}}_{0}$ running the emulated party protocol $\pi_0 = \pi^{\mathsf{des},\rho}_{\mathsf{SA},0}$. Moreover, we have seen in Section 4.6 that protocol π^{in} UC securely and thus stand-alone securely implements the hybrid MPC functionality F^{ρ} from functionality $\mathsf{J}^{\mathsf{des}}$.

Protocol π^{in} relies on a single instance of the functionality $\mathsf{J}^{\mathsf{des}}$ as its only resource. By sequential composition for stand-alone secure protocols as in [Gol04] it follows that protocol $\pi^{\mathsf{in}} \circ \pi^{\mathsf{des},\rho}_{\mathsf{SA}}$ securely implements the hybrid MPC functionality F^{ρ} from an emulated party functionality $\mathsf{P}^{\mathsf{em}}_{0}$ running the emulated party protocol $\pi_0 = \pi^{\mathsf{des},\rho}_{\mathsf{SA},0}$:

By Lemma 4.5.2 for any adversary A we have a SA simulator $\mathsf{S}^{\mathsf{des}}_{\mathsf{SA}}$ for $\pi^{\mathsf{des},\rho}_{\mathsf{SA}}$ such that $\mathsf{S}^{\mathsf{des}}_{\mathsf{SA}}(\mathsf{A})\mathsf{J}^{\mathsf{des}}$ is CO SA indistinguishable from $\pi^{\mathsf{des},\rho}_{\mathsf{SA},\mathcal{H}} \circ \mathsf{P}^{\mathsf{em}}_{0} \circ \mathsf{A}$ We can construct a stand-alone simulator $\mathsf{S}_{\mathsf{SA}} = \mathsf{S}^{\mathsf{in}} \circ \mathsf{S}^{\mathsf{des}}_{\mathsf{SA}}$ by taking the UC simulator S^{in} from Section 4.6.1 and combining it with the simulator $\mathsf{S}^{\mathsf{des}}_{\mathsf{SA}}$. As the UC simulator S^{in} simulates an instance of functionality $\mathsf{J}^{\mathsf{des}}$ to the distinguisher, and as the simulator $\mathsf{S}^{\mathsf{des}}_{\mathsf{SA}}$ expects to interact with functionality $\mathsf{J}^{\mathsf{des}}$, we may directly connect simulator $\mathsf{S}^{\mathsf{des}}_{\mathsf{SA}}$ to the distinguisher (adversary) interfaces of simulator S^{in}. We obtain a simulator $\mathsf{S}_{\mathsf{SA}} = \mathsf{S}^{\mathsf{in}} \circ \mathsf{S}^{\mathsf{des}}_{\mathsf{SA}}$

which renders the ideal model $\mathsf{S}_{\mathsf{SA}}(\mathsf{A}) \circ \mathsf{F}^{\rho}$ stand-alone indistinguishable from the real model $\pi^{\mathsf{in}}_{\mathcal{H}} \circ \pi^{\mathsf{des},\rho}_{\mathsf{SA},\mathcal{H}} \circ \mathsf{P}^{\mathsf{em}}_{0} \circ \mathsf{A}$ for any adversary A.

Thus, we have that $\pi^{\rho}_{\mathsf{SA}} = \pi^{\mathsf{in}} \circ \pi^{\mathsf{des},\rho}_{\mathsf{SA}} \circ \pi^{\mathsf{em}} \circ \pi^{\mathsf{pm}}$ stand-alone securely implements the hybrid MPC functionality F^{ρ} from an n-party network Net^{n}, as claimed.

4.8 Protocols Without Broadcast Channel

We now describe what can be achieved without assuming a BC channel. As our protocol relies on a BC channel, we have to implement one from pairwise secure channels. We make use of the IT secure BC with extended consistency and validity detection $\mathsf{BC}_{\mathsf{extCons}}$ of [FHHW03]. For two thresholds t_v and t_c, where $t_v \leq t_c$ and either $t_v = 0$ or $t_v + 2t_c < n$, $\mathsf{BC}_{\mathsf{extCons}}$ delivers a robust BC for $t \leq t_v$ and a BC with fairness (but without robustness) for $t_v < t \leq t_c$. Actually, $\mathsf{BC}_{\mathsf{extCons}}$ performs a *detectable precomputation* which either establishes a setup for a robust BC (for $t \leq t_v$ always) or aborts with agreement on abort.

For a robustness bound $\rho > 0$ we let $t_v = \rho < \frac{n}{3}$ and $t_c = \lceil \frac{n-t_v}{2} \rceil - 1$. This achieves IT full security (with robustness) for $t \leq \rho$ and IT fair security (no robustness) for $t < \frac{n-\rho}{2}$. Unfortunately these results do not (and cannot) go beyond those of [FHHW03] which they have proven optimal for this case.

However, for robustness bound $\rho = 0$, we let $t_v = \rho = 0$ and $t_c = n$. In this case we achieve IT fair security (no robustness) for $t < \frac{n}{2}$ and CO abort security for $t < n$. This result is new and actually matches the result for $\rho = 0$ according to Theorem 4.3.1 in the case where a BC channel is provided. We refer to π^{ρ} for $\rho = 0$, running with the above BC implementation as π^{0} and have:

Theorem 4.8.1. *Let* F *be an ideal* n*-party functionality and let* $\rho < \frac{n}{2}$ *be a robustness parameter. Let a* CRS *setup and a complete and synchronous network of secure channels (*without *BC channel) be given. Protocol* π^{0} *then implements functionality* F^{ρ} *UC securely against static adversaries corrupting any number* t *of parties. That is,* π^{0} *implements the ideal functionality* F

1. *with IT full security, given that* $t = 0$,
2. *with IT fair security (as formalized by* $\mathsf{F}^{\mathsf{fair}}$*), given that* $t < \frac{n}{2}$*, and*
3. *with CO abort security (as formalized by* F^{ab}*), always.*

4.9 Conclusions

We describe a hybrid secure MPC protocol π^ρ that provides a flexible and optimal trade-off between IT full security (with robustness), IT fair security (no robustness), and CO abort security (no fairness). More precisely, for an arbitrarily chosen robustness parameter $\rho < \frac{n}{2}$, the hybrid-secure MPC protocol π^ρ is IT full secure for $t \leq \rho$, IT fair secure for $t < \frac{n}{2}$, and CO abort secure for $t < n - \rho$ actively and statically corrupted parties. These results are optimal with respect to the bounds stated in [Cle86, Kat06, IKLP06]. On the technical side, we provide a first formal treatment of player emulation in the UC setting.

We prove the UC security of π^ρ in the synchronous secure channels model with broadcast (BC) and a CRS. We also show a simple variation π^ρ_{SA} of protocol π^ρ that relies on [GMW87] instead of [CLOS02] and is stand-alone secure in the synchronous secure channels model with BC *without* a CRS.

Furthermore we discuss the synchronous secure channels model *without* BC. Here we find that for robustness parameter $\rho > 0$ the results of [FHHW03] are already optimal, but for $\rho = 0$ our protocol achieves the same results as in the case where BC is provided, indicating that a BC channel is only helpful if one aims for robustness.

Chapter 5

On the Inexistence of Field-Homomorphic One-Way Permutations

5.1 Introduction

In this chapter we show the inexistence of field-homomorphic one-way permutations (OWPs), for fields of small characteristic or in the non-uniform setting. We discuss implications for constructions of field-homomorphic encryption. Homomorphic encryption is an important tool in non-interactive MPC and server-aided secure computation.

We obtain the desired result on field-homomorphic OWPs, by investigating black-box fields. Black-box fields serve to formalize representation independent, generic algorithms for finite fields. The material presented in this chapter is published in [MR07].

5.1.1 Black-Boxes and Generic Algorithms

Algebraic structures like groups, rings, and fields, and algorithms on them, play a crucial role in cryptography. In order to compute in an algebraic structure one needs a representation of its elements, for instance as

bitstrings. Algorithms that do not exploit any property of the representation are called *generic*. The concept of generic algorithms is of interest for two reasons. First, generic algorithms can be used no matter how the structure is represented, and second, this model allows for significant lower bound proofs for certain computational problems. For instance, Shoup [Sho97] proved a lower bound on the complexity of any generic algorithm for computing discrete logarithms in a finite cyclic group.

Representation-independent algorithms on a given algebraic structure S are best modeled by a *black-box* [BS84, BB99, Mau05], which initially contains some elements of S, describing an instance of the computational problem under consideration. The black-box accepts instructions to perform the operation(s) of S on the values stored in it. The (internal) values are stored in addressable registers and the result of an operation is stored in a new register. The values stored in the black-box are hidden and the only information about these values provided to the outside (an hence to the algorithm) are equalities of stored elements. This models that there is no (need for a) representation of values but that nevertheless one can compute on given values. The equality check provided by the black-box models the trivial property of any (unique) representation that equality is easily checked.[29]

A basic problem in this setting is the *extraction problem*: The black-box contains a secret value x (and possibly also some constants), and the task of the algorithm is to compute x (explicitly).

For example, a cyclic group of prime order p is modeled by a black-box where S is the additive group $\mathbb{Z}_p$ (and which can be assumed to contain the constants 0 and 1 corresponding to the neutral element and the generator, respectively). The discrete logarithm problem is the extraction problem for this black-box. Shoup's result implies that no algorithm can extract x (if uniformly chosen) with fewer than $O(\sqrt{p})$ expected operations. Actually, this many operations are required in expectation to provoke a single collision in the black-box, which is necessary for the algorithm to obtain any information about the content of the black-box. Both the baby-step giant-step algorithm and the Pohlig-Hellman algorithm are generic algorithm which can be described and analyzed in this model.

[29] Note that this model is simpler than Shoup's model which assumes a random representation.

5.1.2 Black-Box Fields and Known Results

If one assumes in the above setting that the black-box not only allows *addition* but also *multiplication* of values modulo p, then this corresponds to a *black-box field* (BBF).

An efficient (non-uniform) algorithm for the extraction problem in $\mathbb{F}_p$ was proposed in [Mau94] (see also [MW99]), where non-uniform means that the algorithm depends on p or, equivalently, obtains a help-string that depends on p. Moreover, the existence of the help-string, which is actually the description of an elliptic curve of smooth order over $\mathbb{F}_p$, depends on a plausible but unproven number-theoretic conjecture.

Boneh and Lipton [BL96] proposed a similar but *uniform* algorithm for the extraction problem in $\mathbb{F}_p$, but its running time is subexponential and the analysis also relies on a related unproven number-theoretic conjecture.

5.1.3 Black-Box Extention Fields

Prime fields differ significantly from extension fields, which is relevant in the context of this work:

In contrast to an extension field $\mathbb{F}_{p^k}$ (for $k > 1$), a prime field $\mathbb{F}_p$ is generated by any non-zero element (for instance 1). Hence there is a unique isomorphism between any two instantiations of $\mathbb{F}_p$ that is given by mapping the 1 of the first instance to the 1 of the second. In particular, there is a unique isomorphism between a BBF over $\mathbb{F}_p$ and any explicit representation of $\mathbb{F}_p$. Therefore in an explicit representation there exists a unique element corresponding to a secret value x inside the black-box, and the extraction problem as stated above is well defined.

As an extension field $\mathbb{F}_{p^k}$ (for $k > 1$) contains non-zero elements that do *not* algebraically generate the entire field, it is not sufficient to give a secret value x inside the black box in order to describe an arbitrary extension field. Rather, the field must be given by a set of elements (generators) in the black-box algebraically generating the field. A vector space basis of $\mathbb{F}_{p^k}$ over $\mathbb{F}_p$ would be a natural choice, but our goal is to make no assumption whatsoever about how the given elements generate the field.

Furthermore, extension fields $\mathbb{F}_{p^k}$ (for $k > 1$) have non-trivial automorphisms, so there is *no unique* isomorphism between a black-box extension field and an explicit representation. Therefore the extraction

problem as originally posed is not well defined for extension fields. We hence formulate a more general problem for extension fields, the *representation problem*: Write a secret x hidden inside the black-box as an algebraic expression in the other elements (generators) given in the black-box.

When an explicit representation of the field is given outside of the black-box (say in terms of an irreducible polynomial of degree k over $\mathbb{F}_p$), then one can also consider the problem of efficiently computing an isomorphism (and its inverse) between this explicitly given field and the BBF.

5.1.4 Contributions

We present an efficient reduction of the representation problem for a finite black-box extension field to the extraction problem for the underlying prime field $\mathbb{F}_p$. If the characteristic p of the field in question is small, or if p is large but an efficient algorithm for the extraction problem for $\mathbb{F}_p$ exists, then this yields an efficient algorithm for the representation problem for the extension field. Under their respective number-theoretic assumptions one can also use the results of [Mau94, BL96, MW99].

Theorem 5.1.1 (Informal version of Theorem 5.3.2)**.** *The representation problem for the finite black-box extension field $\mathbb{F}_{\mathbf{B}}$ of characteristic p is efficiently reducible to the representation problem for $\mathbb{F}_p$. If the characteristic p is small (e.g. $p = 2$) then the representation problem for $\mathbb{F}_{\mathbf{B}}$ is efficiently solvable.*

Furthermore, our algorithms provide an efficiently computable isomorphism between the black-box field and an explicitly represented (outside the black-box) isomorphic copy. If we are given preimages of the generators inside the black-box under some isomorphism from an explicitly represented field into the black-box or if the black-box allows inserting elements from an explicitly represented field, we may even efficiently extract any element from the black-box field, i.e., we can find the element corresponding to an x inside the black-box in the explicit representation.

In particular, these results imply that any problem posed for a black-box field (of small characteristic) can efficiently be transformed into a problem for an explicit field and be solved there using unrestricted (representation-dependent) methods. For example, this implies that computing discrete logarithms in the multiplicative group over a finite field (of small characteristic) is not harder in the black-box setting than in the case where the field is given by an irreducible polynomial.

5.1.5 Cryptographic Significance of Black-Box Fields

A BBF $\mathbb{F}_p$ can be viewed as a black-box group of prime order p, where the multiplication operation of the field corresponds to a Diffie-Hellman oracle; therefore an efficient algorithm for the extraction problem for $\mathbb{F}_p$ corresponds to an efficient generic reduction of the discrete logarithm problem to the computational Diffie-Hellman problem in any group of prime order p (see [Mau94]). So an efficient algorithm for the extraction problem for $\mathbb{F}_p$ provides a security proof for the Diffie-Hellman key agreement protocol [DH76] in any group of order p for which the discrete logarithm problem is hard.[30]

Boneh and Lipton [BL96] gave a second reason why the extraction problem is of interest in cryptography:

Consider the RSA trapdoor one-way permutation (OWP), defined by $x \mapsto x^e \pmod n$. RSA is group-homomorphic: the product of two ciphertexts x^e and x'^e is the ciphertext for their product: $x^e \cdot x'^e = (x \cdot x')^e$. This algebraic property has proven enormously useful in many cryptographic protocols. However, this homomorphic property is only for one operation (i.e., for a group), and a long standing problem in cryptography is to devise a trapdoor one-way permutation that is field-homomorphic, i.e., homomorphic for addition *and* for multiplication. Such a scheme might serve as building block for a field-homomorphic encryption scheme and would have applications in multi-party computation, computation with encrypted data (e.g. server-assisted computation), and possibly other areas in cryptography [SYY99, ALN87, Dom02].

Now, a solution to the extraction problem for $\mathbb{F}_p$ implies an equally efficient attack on any $\mathbb{F}_p$-homomorphic encryption scheme that permits checking the equality of two encrypted elements (which is for example true for any deterministic scheme). Indeed, a black-box field can be regarded as an idealized formulation of a field-homomorphic encryption scheme which allows for equality checks. Any algorithm that succeeds in recovering an "encrypted" element hidden inside the black-box will also break an encryption scheme that allows the same operations. In particular, an efficient algorithm for the extraction problem for $\mathbb{F}_p$ implies the inexistence of a secure $\mathbb{F}_p$-homomorphic one-way permutation.

This generalizes naturally to the extension field case yielding the following corollary to Theorem 5.1.1:

[30] In this context it is not a problem that Maurer's efficient algorithm [Mau94] for the extraction problem for $\mathbb{F}_p$ is non-uniform, because one can construct a Diffie-Hellman group of order p together with the help-string and hence the equivalence really holds.

Corollary 5.1.2. *For fields of small characteristic p (in particular for $\mathbb{F}_{2^k}$) there are no secure field-homomorphic encryption schemes*[31] *that permit equality checks. In particular, there are no field-homomorphic one-way permutations over such fields.*[32]

The same holds even for large characteristic p if we admit non-uniform adversaries under the assumption of [Mau94, MW99].

Our result remains relevant even in the light of recent work by Gentry [Gen09], who proposed a field-homomorphic encryption scheme taking another approach, applying a bootstrapping process to a novel, lattice-based construction. However, it is still interesting to explore the space of possible constructions for field-homomorphic encryption schemes, as under standard lattice assumptions Gentry's scheme is limited to computing circuits of depth at most linear in the size of the public key.

Beyond its cryptographic significance, the representation problem for black-box extension fields is of independent mathematical interest. The representation problem for groups, in particular black-box groups, has been extensively studied [BB99, BS84], inciting interest in the representation problem for other algebraic black-box structures.

5.2 The Representation Problem for Finite Black-Box Fields

5.2.1 Preliminaries on Finite Fields

We assume that the reader is familiar with the basic algebraic concepts of groups, rings, fields, and vector spaces and we summarize a few basic facts about finite fields.

The cardinality of every finite field is a prime power, p^k, where p is called the *characteristic* and k the *extension degree*. There exists a finite

[31] In the public-key case we can efficiently recover the encrypted field element, in the private-key case this is only possible up to isomorphism, as we may have no knowledge of the plaintext field.

[32] One may be led to believe that field-homomorphic one-way permutations cannot exist, since a finite field has only a small number of automorphisms, which can be enumerated exhaustively. However, we assume the target field to be given as a black-box without explicit representation of the elements. As such it is a priori not clear how to find the preimage of a random element.

field for every prime p and every k. Finite fields of equal cardinality are isomorphic, i.e., for each cardinality p^k there is up to isomorphism only one finite field, which allows one to refer to it just as $\mathbb{F}_{p^k}$.

Prime fields $\mathbb{F}_p$ (i.e., $k = 1$) are defined as $\mathbb{Z}_p = \{0, \ldots, p-1\}$ with addition and multiplication modulo p. An extension field $\mathbb{F}_{p^k}$ can be defined as the polynomial ring $\mathbb{F}_p[X]$ modulo an irreducible polynomial $m(X)$ of degree k over $\mathbb{F}_p$. It hence consists of all polynomials of degree at most $k-1$ with coefficients in $\mathbb{F}_p$.

For every $x \in \mathbb{F}_{p^k}$, the p-fold sum of x (i.e., $x+x+\cdots+x$ with p terms), denoted px, is zero: $px = 0$. Moreover, $x^{p^k-1} = 1$ for all $x \neq 0$, as $p^k - 1$ is the cardinality of the multiplicative group of $\mathbb{F}_{p^k}$, which is actually cyclic.

An extension field $\mathbb{F}_{p^k}$ is a vector space over $\mathbb{F}_p$ of dimension k. For appropriate $g \in \mathbb{F}_{p^k}$ there exist bases of the form $(1, g, g^2, \ldots, g^{k-1})$. The only automorphisms of a finite field $\mathbb{F}_{p^k}$ are the Frobenius automorphisms $x \mapsto x^{(p^i)}$ for $i = 0, \ldots, k-1$. In particular, a prime field has no non-trivial automorphisms.

For every ℓ dividing k, there is a subfield $\mathbb{F}_{p^\ell}$ of $\mathbb{F}_{p^k}$. The *trace function* $\mathrm{tr}_{\mathbb{F}_{p^k}/\mathbb{F}_{p^\ell}} : \mathbb{F}_{p^k} \to \mathbb{F}_{p^\ell}$, defined as

$$\mathrm{tr}_{\mathbb{F}_{p^k}/\mathbb{F}_{p^\ell}}(a) = \sum_{i=0}^{(k/\ell)-1} a^{(p^{i\ell})},$$

is a surjective and $\mathbb{F}_{p^\ell}$-linear function [LN97].

5.2.2 The Black-box Model

We make use of the abstract model of computation from [Mau05]: A black-box field $\mathbb{F}_{\mathbf{B}}$ is characterized by a black-box $\mathbf{B}$ which can store an (unbounded number of) values from some finite field $\mathbb{F}_{p^k}$ of known characteristic p but not necessarily known extension degree in internal registers $V_0, V_1, V_2, \ldots$. The first $d+1$ of these registers hold the initial state $I = [g_0, g_1, \ldots, g_d]$ of the black-box. We require the size $d+1$ of the initial state to be at most polynomial in $\mathrm{ld}(|\mathbb{F}_{\mathbf{B}}|)$.

The black-box $\mathbf{B}$ provides the following interface: It takes as input a pair (i, j) of indices and a bit indicating whether addition or multiplication should be invoked. Then it performs the required operation on V_i

and V_j, stores the result in the next free register, say V_ℓ, and reports all pairs of indices (m, n) such that $V_m = V_n$.[33]

Since we only allow performing the field operations $+$ and $\cdot$ on the values of the black box, the black-box field $\mathbb{F}_{\mathbf{B}}$ is by definition the field $\mathbb{F}_{\mathbf{B}} = \mathbb{F}_p[g_0, g_1, \ldots, g_d]$ generated[34] by the elements $g_0, g_1, \ldots, g_d \in \mathbb{F}_{p^k}$ contained in the initial state $I = [g_0, g_1, \ldots, g_d]$ of the black-box.

A black-box field $\mathbb{F}_{\mathbf{B}}$ is thus completely characterized by the

- **public values:** characteristic[35] p, size $d + 1$ of the initial state,
- **secret values:** initial state $I = [g_0, g_1, \ldots, g_d]$ (hidden inside the black-box)

This is probably the most basic yet complete way of describing a finite field. Observe that the field $\mathbb{F}_{p^k}$, the elements of which the black-box can store, does not appear in the characterization. Since no algorithm can compute any value not expressible as an expression in the operators $+$ and $\cdot$, and the elements initially given inside the black-box, we can without loss of generality assume that k is such that $\mathbb{F}_{p^k} \cong \mathbb{F}_{\mathbf{B}}$, where k is unknown, but can be efficiently computed as we shall see later.

Also, the operations "additive inverse" and "multiplicative inverse" and the constants 0 and 1 need not be provided explicitly, since they can be computed efficiently given the characteristic p and the field size $|\mathbb{F}_{\mathbf{B}}| = p^k$: We can compute the additive inverse for an element $a \in \mathbb{F}_{\mathbf{B}}^*$ as $-a = (p-1)a$, and the multiplicative inverse is $a^{-1} = a^{p^k-2}$. Furthermore, $1 = a^{p^k-1}$ for any non-zero a and $0 = pa$ for any a. These expressions can be evaluated efficiently using square-and-multiply techniques.

When discussing the complexity of algorithms on black-box fields, we count each invocation of the black-box as one step. Additionally we will take into account the runtime of computations not directly involving the black-box.

We consider an algorithm to be *efficient* if it runs in time polynomial in the bit-size of a field element, $\mathrm{ld}\,|\mathbb{F}_{\mathbf{B}}|$.[36]

[33] Alternatively, equality checks could also be modeled as an explicit operation which must be called with two indices.

[34] By $\mathbb{F}_p[g_0, g_1, \ldots, g_d]$ we denote the field consisting of all polynomial expressions over $\mathbb{F}_p$ in the generators $g_0, g_1, \ldots, g_d$.

[35] If the characteristic p is small it need not be given but can be recovered in time $O(\sqrt{p})$ using a modified Baby-Step-Giant-Step algorithm [Mau05].

[36] The requirement that the size $d+1$ of the initial state be at most polynomial in $\mathrm{ld}(|\mathbb{F}_{\mathbf{B}}|)$ is imposed so that this makes sense.

5.2.3 The Representation Problem and Related Problems

We now turn to the problems we intend to solve. Let a characteristic p be given and let $\mathbf{B}$ be a black-box with initial state $I = [x, g_1, \ldots g_d]$ consisting of generators $g_1, \ldots g_d$ and a challenge x, where $\mathbb{F}_{\mathbf{B}} = \mathbb{F}_p[x, g_1, \ldots g_d]$.We then consider the following problems:

Definition 5.2.1 (Representability Problem, Representation Problem)**.** We call x *representable* (in the generators $g_1, \ldots g_d$) if $x \in \mathbb{F}_p[g_1, \ldots g_d]$. The problem of deciding whether $x \in \mathbb{F}_p[g_1, \ldots g_d]$ is called the *representability problem*. If x is representable, then finding a multi-variate polynomial $q \in \mathbb{F}_p[X_1, \ldots, X_d]$ such that $x = q(g_1, \ldots, g_d)$ is called the *representation problem*. ◊

We proceed to discuss two problems that are closely related to the representation problem. First, we state a generalization of the extraction problem, defined in [Mau05], that is applicable to all finite black-box fields. To do so, we need to specify an isomorphism ϕ from the black-box to some explicitly given field K. This is necessary for the extraction problem to be well-defined, because in contrast to prime fields there are many isomorphisms between two isomorphic extension fields.

Definition 5.2.2 (Extraction Problem)**.** Let K be an explicitly given field (e.g. by an irreducible polynomial) such that $K \cong \mathbb{F}_{\mathbf{B}}$. Let the images $\phi(g_1), \ldots, \phi(g_d)$ of the generators $g_1, \ldots, g_d$ under some isomorphism $\phi : \mathbb{F}_{\mathbf{B}} \to K$ be given. The *extraction problem* is to compute $\phi(x)$.[37] ◊

Remark 5.2.3. Note that an efficient solution to the representation problem implies an efficient solution to the extraction problem. The expression $q(g_1, \ldots, g_d)$ returned as a solution to the representation problem can simply be evaluated over K, substituting $\phi(g_i)$ for g_i ($i = 1, \ldots, d$), which yields $\phi(x)$:

$$q(\phi(g_1), \ldots, \phi(g_d)) = \phi(q(g_1, \ldots, g_d)) = \phi(x).$$

Finally consider an efficient but representation-dependent algorithm A solving some problem Q on a finite field K (where the algorithm A requires for instance that the field K is given by an irreducible polynomial). We are interested if the existence of such an algorithm A generally

[37] The extraction problem also makes sense if the isomorphism ϕ is given in another fashion. For example, the black-box might offer an operation that allows inserting elements from an explicitly given field K. This would for instance correspond to a field-homomorphic one-way permutation.

implies the existence of a generic algorithm for the problem Q of comparable efficiency. More specifically, we are interested in algorithms Φ and Φ^{-1} efficiently computing an *arbitrary* isomorphism $\phi : \mathbb{F}_{\mathbf{B}} \to K$ and its inverse ϕ^{-1}, yielding a generic solution $\Phi^{-1} \circ A \circ \Phi$ to the problem Q. That is the algorithm Φ maps an $x \in \mathbb{F}_{\mathbf{B}}$ to K by solving the extraction problem with respect to ϕ. The inverse map Φ^{-1} on the other hand maps a field element $x' \in K$ into the black box field $\mathbb{F}_{\mathbf{B}}$ by means of constructing $\phi^{-1}(x')$ from the generators inside the black-box using the field operations. These two algorithms can then be chained together with the original, representation dependent algorithm A, yielding a black-box, representation independent algorithm $\Phi^{-1} \circ A \circ \Phi$. Hence we consider the following problem:

Definition 5.2.4 (Isomorphism Problem)**.** Let K be an explicitly given field such that $K \cong \mathbb{F}_{\mathbf{B}}$. The *isomorphism problem* consists of computing an (arbitrary but fixed) isomorphism $\phi : \mathbb{F}_{\mathbf{B}} \to K$ and its inverse ϕ^{-1} for arbitrary elements of K and $\mathbb{F}_{\mathbf{B}}$. $\Diamond$

In the following we will exhibit an efficient reduction from the representation problem for any finite field to the representation problem for the underlying prime field. Moreover, our solution to the representation problem will also yield an explicitly given field (by an irreducible polynomial) $\mathbb{F}_{p^k} \cong \mathbb{F}_{\mathbf{B}}$ with an efficient solution to the isomorphism problem for $\mathbb{F}_{p^k}$ and $\mathbb{F}_{\mathbf{B}}$. This allows to solve any problem posed on the black-box field $\mathbb{F}_{\mathbf{B}}$ in the explicitly given field $\mathbb{F}_{p^k}$ using the corresponding algorithms.

5.2.4 The Representation Problem for $\mathbb{F}_p$

First, we shall see that the representation, extraction and isomorphism problems are equivalent when the black-box field $\mathbb{F}_{\mathbf{B}}$ is isomorphic to some prime field $\mathbb{F}_p$:

Lemma 5.2.5. *Let $\mathbb{F}_{\mathbf{B}} \cong \mathbb{F}_p$ be a BBF with initial state $I = [x, g_1, \ldots, g_d]$. Then the representation, extraction and isomorphism problems are efficiently reducible to one another.*

Proof. Note that there is a unique isomorphism $\phi : \mathbb{F}_{\mathbf{B}} \to \mathbb{F}_p$. Furthermore, as $\mathbb{F}_{\mathbf{B}} \cong \mathbb{F}_p$, there must be a $g_i \neq 0$ ($i \in \{1, \ldots, d\}$). This g_i can be efficiently found by checking the inequality $g_i + g_i \neq g_i$ and the constant

1 can be efficiently computed inside the black-box as g_i^{p-1} using square-and-multiply techniques.

Reduction extraction to representation: see Remark 5.2.3.

Reduction isomorphism to extraction: A solution to the extraction problem yields an efficient algorithm computing the isomorphism ϕ. The inverse ϕ^{-1} can be efficiently computed using square-and-multiply techniques, constructing $\phi^{-1}(a)$ for $a \in \mathbb{F}_p$ as a sum of 1s inside the black-box. This solves the isomorphism problem.

Reduction representation to isomorphism: A solution to the isomorphism problem yields an efficient algorithm computing the isomorphism ϕ. Then we have $\phi(x)g_i^{p-1}$ as a solution to the representation problem.

□

Note that solving the extraction problem for a black-box field $\mathbb{F}_{\mathbf{B}} \cong \mathbb{F}_p$ with initial state $V^1 = [x]$ amounts to solving the discrete logarithm problem for a group of order p (given as a black-box) for which a Diffie-Hellman oracle is given. The following results are known:

Lemma 5.2.6 ([Mau94])**.** *There exists a non-uniform algorithm that, under a (plausible) number-theoretic conjecture, solves the extraction (representation, isomorphism) problem for a black-box field $\mathbb{F}_{\mathbf{B}} \cong \mathbb{F}_p$ in time polynomial in $\mathrm{ld}(p)$, and with a polynomial (in $\mathrm{ld}(p)$) amount of advice depending on p.*

Lemma 5.2.7 ([BL96])**.** *There exists a (uniform) algorithm that, under a (plausible) number-theoretic conjecture, solves the extraction (representation, isomorphism) problem for a black-box field $\mathbb{F}_{\mathbf{B}} \cong \mathbb{F}_p$ in time subexponential in $\mathrm{ld}(p)$.*

For the remainder of this work we will only concern ourselves with reducing other problems to the representation problem for $\mathbb{F}_p$. The reader may generally assume that p is small, such that the representation problem for $\mathbb{F}_p$ is easy to solve.

5.2.5 The Representation Problem for $\mathbb{F}_{p^k}$ for a given $\mathbb{F}_p$-Basis

Before we proceed to the general case, we first investigate the simpler case where the initial state of the black-box $\mathbf{B}$ is $I = [x, b_1, \ldots, b_k]$, and $b_1, \ldots, b_k$ form a basis of $\mathbb{F}_{\mathbf{B}}$ as $\mathbb{F}_p$-vector space. We efficiently reduce this problem to the representation problem for $\mathbb{F}_p$ discussed in Section 5.2.4.

Lemma 5.2.8. *The representation problem for a black-box field $\mathbb{F}_{\mathbf{B}}$ of characteristic p with initial state $I = [x, b_1, \ldots, b_k]$, where $b_1, \ldots, b_k$ form an $\mathbb{F}_p$-basis of $\mathbb{F}_{\mathbf{B}}$, is efficiently reducible to the representation problem for $\mathbb{F}_p$.*

Proof. The proof relies on the well-known dual basis theorem (see e.g. [LN97]): For any $\mathbb{F}_p$-basis $\{b_1, \ldots, b_k\}$ of $\mathbb{F}_{p^k}$ there exists a dual basis $\{c_1, \ldots, c_k\}$ with the property that $\mathrm{tr}_{\mathbb{F}_{p^k}/\mathbb{F}_p}(c_i b_j) = \delta_{ij}$, where δ_{ij} designates the Kronecker-Delta. We calculate the dual basis $\{c_1, \ldots, c_k\}$ for the basis $\{b_1, \ldots, b_k\}$ inside the black-box. This can be done efficiently by employing formal derivatives as in [Lan02, p. 287, Prop. 5.5] or by means of the possibly less elegant but elementary solution described below:

We write the elements of the dual basis as $c_i = \sum_{l=1}^{k} \alpha_{il} b_l$. Furthermore, let $A = (\alpha_{il})_{i,l=1,\ldots,k}$ be the coefficient matrix, $B = (\mathrm{tr}_{\mathbb{F}_{p^k}/\mathbb{F}_p}(b_l b_j))_{l,j=1,\ldots,k}$ the trace matrix, and I_k the identity matrix. Then the definition of the dual basis yields a matrix equation $AB = I_k$. Traces can be computed efficiently inside the black-box using square-and-multiply techniques, so the trace matrix B can be efficiently computed inside the black-box. Since B always has full rank [LN97], the matrix equation $AB = I_k$ can be solved for the α_{il} using Gaussian elimination (inside the box $\mathbf{B}$).

As the characteristic p and the exponent k are known, we can efficiently compute additive and multiplicative inverses (see Section 5.2.2). Solving for the k^2 unknowns in the matrix A using Gaussian elimination is efficient, and requires only field operations and equality checks. Hence it can be performed in the black-box and we can efficiently compute the dual basis elements c_i inside the black-box.

To represent the challenge x in the basis $\{b_1, \ldots, b_k\}$, we now calculate $\xi_i = \mathrm{tr}_{\mathbb{F}_{p^k}/\mathbb{F}_p}(c_i x) \in \mathbb{F}_p$ inside the black-box and have $x = \sum_{i=1}^{k} \xi_i b_i$ by the dual basis property. We use an oracle $\mathcal{O}$ that solves the representation problem for $\mathbb{F}_p$ (possibly instantiated according to Section 5.2.4) to extract the ξ_i from the black box, obtaining the required representation of x in the given generators (basis) $\{b_1, \ldots, b_k\}$. □

5.3 The Representation Problem for $\mathbb{F}_{p^k}$ for Arbitrary Generators

Now we turn to the general case, where a black-box field $\mathbb{F}_{\mathbf{B}}$ of characteristic p is not necessarily given by a basis, but by an arbitrary generating

set $\{g_1, \ldots, g_d\}$ which generates $\mathbb{F}_{\mathbf{B}}$ as $\mathbb{F}_p$-algebra.

5.3.1 Main Theorem

Before we get to our main result, we first discuss the representability problem.

Lemma 5.3.1. *The representability problem for a black-box field $\mathbb{F}_{\mathbf{B}}$ of characteristic p with initial state $I = [x, g_1, \ldots, g_d]$ can be solved efficiently and the extension degree k such that $\mathbb{F}_{\mathbf{B}} \cong \mathbb{F}_{p^k}$ can be found efficiently.*

Proof. We need to determine efficiently whether x is representable in the generators $g_1, \ldots, g_d$ and then find k such that $\mathbb{F}_{\mathbf{B}} \cong \mathbb{F}_{p^k}$. To this end we first determine the size $k_i := k(g_i) := |\mathbb{F}_p[g_i]|$ of the subfield $\mathbb{F}_p[g_i] \leq \mathbb{F}_{\mathbf{B}}$ of the black-box field $\mathbb{F}_{\mathbf{B}}$ generated by g_i, for $i = 1, \ldots, d$. We have

$$k_i := k(g_i) = \min\{j \in \mathbb{N} : g_i = g_i^{p^j}\} \tag{5.1}$$

by the properties of the Frobenius homomorphism $y \mapsto y^p$ [LN97]. Equation (5.1) can be evaluated efficiently using square-and-multiply.

Now the field element x is representable in the generators $g_1, \ldots, g_d$ if and only if $x \in \mathbb{F}_p[g_1, \ldots, g_d]$ or, equivalently, $\mathbb{F}_p[x] \leq \mathbb{F}_p[g_1, \ldots, g_d]$. But the field $\mathbb{F}_p[g_1, \ldots, g_d]$ generated by $g_1, \ldots, g_d$ is isomorphic to the smallest field $\mathbb{F}_{p^{k'}}$ where $k' = \operatorname{lcm}_{i=1}^{l}(k_i)$ that contains all the $\mathbb{F}_{p^{k_i}}$. Hence x is representable in the generators $g_1, \ldots, g_d$ if and only if $k(x) \mid k'$. Moreover, independently of the representability of x we have $k = \operatorname{lcm}(k(x), k')$. □

We can now state our main result, an efficient reduction from the representation problem for an extension field to the representation problem for the underlying prime field:

Theorem 5.3.2. *The representation problem for the black-box field $\mathbb{F}_{\mathbf{B}}$ of characteristic p with initial state $I = [x, g_1, \ldots, g_d]$ (not necessarily a basis) such that x is representable in $g_1, \ldots, g_d$ is efficiently reducible to the representation problem for $\mathbb{F}_p$.*

We shall see later that from this theorem we can also obtain efficient reductions of the extraction and isomorphism problems to the representation problem for the underlying prime field $\mathbb{F}_p$.

5.3.2 Proof of Theorem 5.3.2

By assumption, the challenge x is representable in the generators $g_1, \ldots, g_d$. We will show how to efficiently generate a $\mathbb{F}_p$-power-basis $\{g^0, g^1, \ldots, g^{k-1}\}$ for $\mathbb{F}_{\mathbf{B}}$ inside the black-box. The representation problem can then be efficiently reduced to the representation problem for $\mathbb{F}_p$ using Lemma 5.2.8.[38]

Algorithm 1 returns an $\mathbb{F}_p$-power-basis for $\mathbb{F}_{\mathbf{B}}$ by computing an element $g \in \mathbb{F}_{\mathbf{B}}$ (a generator), such that $\mathbb{F}_p[g] = \mathbb{F}_{p^k}$. To this end Algorithm 1 iterates over the generators $g_1, \ldots, g_d$, checking if the current g_i is already contained in $\mathbb{F}_p[g]$ for the current g.[39] If not, Algorithm 1 invokes the algorithm $\texttt{combine\_gen}(g, g_i)$ to obtain a new g (which we call g' for now) such that $\mathbb{F}_p[g'] = \mathbb{F}_p[g, g_i]$. Clearly, $\mathbb{F}_p[g] = \mathbb{F}_p[g_1, \ldots, g_d]$ when the algorithm terminates, and hence $\{g^0, g^1, \ldots, g^{k-1}\}$ is a $\mathbb{F}_p$-power-basis for $\mathbb{F}_p[g_1, \ldots, g_d] = \mathbb{F}_{\mathbf{B}}$.

Algorithm 1 Compute power-basis

```
1: g := 1
2: m := 1
3: for i = 1 to d do
4:    if k_i ∤ m then
5:       m := lcm(m, k_i)
6:       g := combine_gen(g, g_i)
7:    end if
8: end for
9: return power basis {g^0, g^1, ..., g^{k-1}}
```

As g is computed inside the black-box from the initially given generators $g_1, \ldots, g_d$ using only field operations, a representation $q'(g_1, \ldots, g_d) = g$ of g (and therefore of all basis elements) in the generators $g_1, \ldots, g_d$ is known. Now Lemma 5.2.8 gives a representation

[38] One might suspect that the $\{g_i^j\}_{i=1,\ldots,d;j=1,\ldots,k}$ already generate $\mathbb{F}_{\mathbf{B}}$ as an $\mathbb{F}_p$-vector space. However, this is not the case. As an example, take $\mathbb{F}_{2^6}$. Then we can find generators $g_2 \in \mathbb{F}_{2^2} \subset \mathbb{F}_{2^6}$ and $g_3 \in \mathbb{F}_{2^3} \subset \mathbb{F}_{2^6}$ such that $\mathbb{F}_2[g_2, g_3] = \mathbb{F}_{2^6}$. But $g_i^j \in \mathbb{F}_{2^i}$, so the $\mathbb{F}_p$-vector space V generated by $\{g_i^j\}$ has dimension $\dim_{\mathbb{F}_2} V \leq \dim_{\mathbb{F}_2} \mathbb{F}_{2^2} + \dim_{\mathbb{F}_2} \mathbb{F}_{2^3} = 5 < 6 = \dim_{\mathbb{F}_2} \mathbb{F}_{2^6}$.

[39] Note that the number of generators g_i appearing in the representation of the generator g (and thereby the representation of x) could be reduced by considering only the generators g_i corresponding to the maximal elements in the lattice formed by the k_i under the divisibility relation (these suffice to generate the entire field $\mathbb{F}_{\mathbf{B}}$). For ease of exposition we do not do this.

$q''(g^0, g^1, \ldots, g^{k-1}) = x$ of the challenge x in the basis elements, so a representation $q(g_1, \ldots, g_d) = x$ of x in the generators $g_1, \ldots, g_d$ can be recovered by substitution:

$$\begin{aligned} q(g_1, \ldots, g_d) &= q''(g^0, g^1, \ldots, g^{k-1}) \\ &= q''(q'(g_1, \ldots, g_d)^0, q'(g_1, \ldots, g_d)^1, \ldots, q'(g_1, \ldots, g_d)^{k-1}) \end{aligned}$$

Algorithm 1 is obviously efficient if the algorithm `combine_gen` is efficient. So, to complete the proof of Theorem 5.3.2, we only need to provide an algorithm `combine_gen`(a, b) that, given two elements $a, b \in \mathbb{F}_{\mathbf{B}}$, efficiently computes a generator g such that $\mathbb{F}_p[g] = \mathbb{F}_p[a, b]$.

Algorithm 2 `combine_gen`(a, b)

1: find k'_a , k'_b such that
   - $k'_a \mid k(a)$, $k'_b \mid k(b)$,
   - $\gcd(k'_a, k'_b) = 1$,
   - $\operatorname{lcm}(k'_a, k'_b) = \operatorname{lcm}(k(a), k(b))$

2: find $a' \in \mathbb{F}_p[a]$ and $b' \in \mathbb{F}_p[b]$ such that $k(a') = k'_a$ and $k(b') = k'_b$
3: **return** $a' + b'$

Claim 5.3.3. *Given two elements $a, b \in \mathbb{F}_{\mathbf{B}}$, the algorithm `combine_gen`$(a, b)$ efficiently computes a generator g such that $\mathbb{F}_p[g] = \mathbb{F}_p[a, b]$.*

Proof. We analyze algorithm `combine_gen`(a, b) step by step:

Step 1 can be performed in time polynomial in k (where $p^k = |\mathbb{F}_{\mathbf{B}}|$), and hence in $\operatorname{ld}(|\mathbb{F}_{\mathbf{B}}|)$, by factoring $k(a)$ and $k(b)$ (which both divide k).[40]

Step 2 relies on the following lemma [Len05]:

Lemma 5.3.4. *Let $M \geq L \geq K$ be a tower of finite fields and let $b_1, \ldots, b_n$ be a K-basis of M. Then $\{\operatorname{tr}_{M/L}(b_1), \ldots, \operatorname{tr}_{M/L}(b_n)\}$ contains a K-basis of L.*

[40] Bach and Shallit [BS96, Section 4.8] give a much more efficient algorithm for computing such values k'_a, k'_b of complexity $O((\operatorname{ld} k(a)k(b))^2)$.

Proof. From [LN97, 2.23(iii)] we know that $\mathrm{tr}_{M/L} : M \to L$ is L-linear and surjective. Hence for all $d \in L$ there exists an $c \in M$ such that $\mathrm{tr}_{M/L}(c) = d$. Since $b_1, \ldots, b_n$ form a K-basis of M, the element $c \in M$ can be expressed as $c = \sum_{i=1}^{n} \gamma_i b_i$ where $\gamma_i \in K$ ($i = 1, \ldots, n$). Hence using the L-linearity of $\mathrm{tr}_{M/L}$ we have

$$d = \mathrm{tr}_{M/L}(c) = \mathrm{tr}_{M/L}(\sum_{i=1}^{n} \gamma_i b_i) = \sum_{i=1}^{n} \gamma_i \,\mathrm{tr}_{M/L}(b_i).$$

As we can represent every $d \in L$ by a K-linear combination in $\{\mathrm{tr}_{M/L}(b_1), \ldots, \mathrm{tr}_{M/L}(b_n)\}$, this set must contain a K-basis of L. □

As we know k'_a and $k(a)$ from Step 1, and using the fact that the elements $\{a^i : i = 0, \ldots, k(a)-1\}$ form an $\mathbb{F}_p$-basis of $\mathbb{F}_p[a]$, we can compute the set $\{\mathrm{tr}_{\mathbb{F}_p[a]/\mathbb{F}_{p^{k'_a}}}(a^i) : i = 0, \ldots, k(a) - 1\}$ in time $O(k^3 \,\mathrm{ld}(p))$, which contains by the lemma above an $\mathbb{F}_p$-basis of $\mathbb{F}_{p^{k'_a}}$.

The following claim is from [BvzGL01, Lemma 6.2]. For completeness we provide a short proof sketch.

Claim 5.3.5. *Any $\mathbb{F}_p$-basis of an extension field $\mathbb{F}_{p^\ell}$ contains a basis element a' such that $\mathbb{F}_{p^\ell} = \mathbb{F}_p[a']$.*

sketch. The $\mathbb{F}_p$-dimension of the span of all proper subfields of $\mathbb{F}_{p^\ell}$ can be computed by application of the inclusion-exclusion principle (first adding the dimensions of all maximal subfields, then subtracting the dimensions of their intersections, then adding the dimensions of the intersections of the intersections, and so on). Using the Möbius function μ and the Euler function φ we can hence write the $\mathbb{F}_p$-dimension of the span of all proper subfields of $\mathbb{F}_{p^\ell}$ as $-\sum_{d|\ell, d\neq\ell} \mu(\ell/d)d = \ell - \varphi(\ell) < \ell$. As the $\mathbb{F}_p$-dimension of the span of all proper subfields of $\mathbb{F}_{p^\ell}$ is smaller then the $\mathbb{F}_p$-dimension ℓ of $\mathbb{F}_{p^\ell}$, there must be a basis element a' which is not contained in any proper subfield of $\mathbb{F}_{p^\ell}$, and therefore $\mathbb{F}_{p^\ell} = \mathbb{F}_p[a']$. □

By the claim above there is a basis element a', that generates $\mathbb{F}_{p^{k'_a}}$, i.e. $\mathbb{F}_{p^{k'_a}} = \mathbb{F}_p[a']$:

$$\exists a' \in \{\mathrm{tr}_{\mathbb{F}_p[a]/\mathbb{F}_{p^{k'_a}}}(a^i) : i = 0, \ldots, k(a) - 1\} \; : \quad k(a') = k'_a.$$

By checking this property for all candidate elements in $\{\mathrm{tr}_{\mathbb{F}_p[a]/\mathbb{F}_{p^{k'_a}}}(a^i) : i = 0, \ldots, k(a) - 1\}$ we find the generator a' in time $O(k^3 \,\mathrm{ld}(p))$. Analogously we may determine b' such that $k(b') = k'_b$.

Step 3. To complete the analysis of the algorithm `combine_gen`(x, y), it remains to show that given a', b' from Step 2, we have $\mathbb{F}_p[a'+b'] = \mathbb{F}_p[a, b]$. Since $\mathrm{lcm}(k(a'), k(b')) = \mathrm{lcm}(k(a), k(b))$ by Step 1, we have $\mathbb{F}_p[a', b'] = \mathbb{F}_p[a, b]$, so it only remains to show that $\mathbb{F}_p[a' + b'] = \mathbb{F}_p[a', b']$. We have $\mathbb{F}_p[a', b'] = \mathbb{F}_p[a', a'+b'] = \mathbb{F}_p[a'+b', b']$ and $\gcd(k(a'), k(b')) = 1$, therefore

$$\begin{aligned}\mathrm{lcm}(k(a'), k(b')) &= \mathrm{lcm}(k(a'), k(a' + b')) \\ &= \mathrm{lcm}(k(a' + b'), k(b')) = k(a')k(b').\end{aligned}$$

It is easy to see that then $k(a' + b') = k(a')k(b')$ holds, and therefore $\mathbb{F}_p[a' + b'] = \mathbb{F}_p[a, b]$, as required. □

5.3.3 Implications of Theorem 5.3.2

From Theorem 5.3.2 and Remark 5.2.3 we obtain the following corollary:

Corollary 5.3.6. *The extraction problem for any BBF $\mathbb{F}_{\mathbf{B}}$ of characteristic p is efficiently reducible to the representation problem for $\mathbb{F}_p$.*

The extraction problem asks for the computation of an isomorphism $\phi : \mathbb{F}_{\mathbf{B}} \to K$. Note that the computation of ϕ^{-1} also reduces efficiently to the representation problem for $\mathbb{F}_p$, because we can efficiently obtain a power-basis $\{g^0, g^1, \ldots, g^{k-1}\}$ inside the black-box, as in the proof of Theorem 5.3.2. From this basis we can then compute the basis $\{\phi(g^0), \phi(g^1), \ldots, \phi(g^{k-1})\}$ for K. Hence the isomorphism ϕ^{-1} can be simply and efficiently computed by basis representation.

Corollary 5.3.7. *Let $\mathbb{F}_{\mathbf{B}}$ be a BBF of characteristic p and K some explicitly given field (in the sense of [Len91]) such that $K \cong \mathbb{F}_{\mathbf{B}}$. Then the isomorphism problem for $\mathbb{F}_{\mathbf{B}}$ and K can be efficiently reduced to the representation problem for $\mathbb{F}_p$.*

Proof. We show that it is possible to efficiently find a field $K' \cong \mathbb{F}_{\mathbf{B}}$ that is explicitly given by an irreducible polynomial, such that the isomorphism problem for $\mathbb{F}_{\mathbf{B}}$ and K' efficiently reduces to the representation problem for $\mathbb{F}_p$. The corollary then follows from [Len91], which states that the isomorphism problem for two explicitly given finite fields can be solved efficiently.

So, let an oracle $\mathcal{O}$ for the representation problem over $\mathbb{F}_p$ be given. As in the proof of Theorem 5.3.2 we efficiently compute a power-basis $\{g^0, g^1, \ldots, g^{k-1}\}$ inside the black-box. By Lemma 5.2.8 we compute a

representation $q(g^0, g^1, \ldots, g^{k-1}) = g^k$ of g^k in the basis elements. Note that the minimal polynomial $f_g \in \mathbb{F}_p[X]$ of g over $\mathbb{F}_p$ is then exactly $f_g(X) = X^k - q(X^0, X^1, \ldots, X^{k-1})$. Let $K' = \mathbb{F}_p[X]/(f_g)$. Then the required isomorphisms ϕ and ϕ^{-1} are efficiently given by basis representation. □

5.4 Conclusions

We have shown that, given an efficient algorithm for the representation problem for $\mathbb{F}_p$, we can solve the representability, representation, extraction and isomorphism problems for a black-box extension field $\mathbb{F}_{\mathbf{B}} \cong \mathbb{F}_{p^k}$ in polynomial time. We achieve this by efficiently constructing (in the generators) an $\mathbb{F}_p$-power-basis $\{g^0, g^1, \ldots, g^{k-1}\}$ for the black-box field $\mathbb{F}_{\mathbf{B}}$ inside the black-box, which is interesting in its own right.

For small characteristic p we can immediately solve the above problems efficiently, as in this case solving the representation problem for $\mathbb{F}_p$ (e.g. using Baby-Step-Giant-Step) is easy.

As a consequence, field-homomorphic one-way permutations over fields of small characteristic p, in particular over $\mathbb{F}_{2^k}$, do not exist, because such a function would constitute an instantiation of a black-box field[41] and could be efficiently inverted using the solution to the extraction problem given above. This implies that over fields of small characteristic there can be no field-homomorphic analogue to the group-homomorphic RSA encryption scheme, which constitutes a group-homomorphic trapdoor one-way permutation.

For the same reason, deterministic and even probabilistic field-homomorphic encryption schemes (both private-[42] and public-key) over fields of small characteristic p, in particular over $\mathbb{F}_{2^k}$, cannot be realized, if they allow for checking the equality of elements. This is unfortunate because such schemes could have interesting applications in multi-party computation and computation with encrypted data (e.g. server-assisted computation) [SYY99, ALN87, Dom02]. For instance we might be interested in handing encrypted field elements to a computing facility and

[41] Instead of generators we have here the possibility to "insert" elements of an explicitly given field into the "black-box" of the image of the function.

[42] This result requires Theorem 5.3.2 whereas the results above already follow from Lemma 5.2.8. Also, note that in the private-key case it is only possible to recover encrypted field elements up to isomorphism, as we may have no knowledge of the plaintext field.

having it compute some (known) program on them. If the encryption permits equality checks, the computing facility can efficiently recover the field elements up to isomorphism.

A randomized field-homomorphic encryption scheme was recently proposed by Gentry [Gen09]. Gentry takes a different approach, applying a bootstrapping process to a novel, lattice-based construction. As such he avoids the impossibility result which we obtain for constructions based on field-homomorphic one-way permutations. However, it is still interesting to explore the space of possible constructions for field-homomorphic encryption schemes, as under standard lattice assumptions Gentry's scheme is limited to computing circuits of depth at most linear in the size of the public key. Thus, our contribution remains relevant.

Furthermore, a polynomial-time solution to the isomorphism problem implies that any problem posed on a black-box field (i.e., computing discrete logarithms over the multiplicative group) can be efficiently transferred to an explicitly represented field, and be solved there using possibly representation-dependent algorithms (e.g. the number field sieve). The solution can then efficiently be transferred back to the black-box field. So any representation-dependent algorithm for finite fields is applicable (in the case of small characteristic) to black-box fields. For example, computing discrete logarithms in the multiplicative group over a finite field is no harder in the black-box setting than if the field is given explicitly by an irreducible polynomial.

Of course these conclusions do apply not only to fields of small characteristic p, but to any scenario where we can efficiently solve the representation problem for the underlying prime field $\mathbb{F}_p$. Hence we obtain subexponential-time solutions to the above problems under a plausible number-theoretic conjecture applying the work of Boneh and Lipton [BL96] for solving the representation problem for $\mathbb{F}_p$. Furthermore we can, under a plausible number-theoretic conjecture, solve the problems above efficiently, even for large characteristic p, if we are willing to admit non-uniform solutions (solutions that require a polynomial amount of advice depending on the characteristic p) using an algorithm by Maurer [Mau94] for solving the representation problem for $\mathbb{F}_p$.

Compared to the case of small characteristic, the situation for fields of large characteristic is then more complex, because the only known efficient algorithm for solving the representation problem for $\mathbb{F}_p$ is non-uniform [Mau94, MW99], i.e. it requires a help-string that depends on p.

When considering homomorphic encryption and homomorphic one-way permutations, this means that our impossibility results hold for cases where a malicious party may fix the characteristic p. In this case the attacker can generate p along with the required help-string to break the scheme. On the other hand our impossibility results do not apply if the characteristic p cannot be determined by the attacker, for instance because it is generated by a trusted party.

It remains an open problem to resolve this issue by providing an efficient *uniform* algorithm for the representation problem for $\mathbb{F}_p$, or by proving the inexistence thereof.

Chapter 6

Concluding Remarks

In this thesis we explore the limitations of secure function evaluation (SFE) and multi-party computation (MPC). Motivated by broad impossibility results for general MPC with full information-theoretic (IT) security in presence of a corrupted majority, we follow two lines of inquiry:

1. If *general* MPC guaranteeing full IT security against $t \geq \frac{n}{2}$ corrupted parties is impossible, then what *specific* functions can still be computed in this setting?

2. If general MPC guaranteeing *full IT security* against $t \geq \frac{n}{2}$ corrupted parties is impossible, is it at least possible to obtain a graceful degradation of security properties? Can we devise a protocol that is fully IT security in case of few corruptions, but still provides CO security without fairness in case of many corruptions?

Additionally we discuss homomorphic encryption as a tool for non-interactive protocols.

Our contribution is thus three-fold:

1. We provide a combinatorial characterization of the functions IT securely computable in presence of active, semi-honest and passive adversaries in the authenticated channels model with broadcast using constant round protocols.

We then derive a characterizations of the functions computable with long-term (LT) security in presence of active adversaries in the same model or in a very realistic, internet-like communication model, where a public-key infrastructure and insecure channels are provided.

Long-term security is a very interesting paradigm: The class of LT securely computable functions is strictly larger than the class of IT securely computable functions in presence of active adversaries. At the same time the loss of security is marginal. A LT secure protocol relies on unproven computational assumptions only for the duration of protocol execution and provides unconditional security after protocol termination. This is desirable, as the problem with CO assumptions is not so much that these could be unjustified right now, but rather that concrete CO assumptions could *eventually* be broken by an adversary whose power increases over time, e.g. due to new technical or algorithmic developments.

2. We construct a hybrid-secure MPC protocol that allows for graceful degradation of security with an increasing number of corrupted parties. Our protocol π_ρ is optimal under the bounds of Kilian [Kil00], Cleve [Cle86], Ishai et al. [IKLP06] and Katz [Kat06]: For any fixed robustness parameter $\rho < \frac{n}{2}$ protocol π_ρ simultaneously provides robust IT security in the presence of $t \leq \rho$ actively corrupted parties, fair IT security (no robustness) for $t < \frac{n}{2}$, and CO security with agreement on abort (no robustness) for $t < n - \rho$.

3. We prove the inexistence of field-homomorphic one-way permutations. Our result holds in the uniform sense only for fields of small characteristic but applies to all fields when the adversary may choose the characteristic or is provided with a polynomial amount of advise depending on it.

   We find that constructions for field-homomorphic encryption along the lines of group-homomorphic schemes like RSA are impossible, since these types of construction generally rely on homomorphic one-way permutations.

We conclude by pointing out a number of interesting open questions that may inspire further research:

- We provide a characterization of functions computable by constant round protocols in presence of active, passive, and semi-honest ad-

versaries. We conjecture that our characterization applies regardless of the round complexity of the protocol. The proof, however, is still open, except for special cases.

- We discuss hybrid secure MPC in the secure channels model with or without broadcast. Still open is the setting where instead of a broadcast channel only signature setup is available, which may be inconsistent or subject to signature forgery.

- Gentry [Gen09] recently proposed a field-homomorphic public-key encryption scheme that allows evaluating circuits of depth linear in the size of the public key. For evaluating circuits of depth independent of the size of the public key, Gentry has to resort to non-standard assumptions. We prove the infeasibility of a particular approach to the problem. But, providing a field-homomorphic public-key encryption scheme based on standard assumptions where the size of the public key is independent of the circuit depth it can evaluate remains an open problem.

Bibliography

[ALN87] Niv Ahituv, Yeheskel Lapid, and Seev Neumann. Processing encrypted data. *Communications of the ACM*, 30(9):777–780, 1987.

[BB99] László Babai and Robert Beals. A polynomial-time theory of black box groups I. *London Mathematical Society Lecture Note Series*, 260:30–64, 1999.

[BCMS99] Gilles Brassard, Claude Crépeau, Dominic Mayers, and Louis Salvail. Defeating classical bit commitments with a quantum computer. Los Alamos preprint archive quant-ph/9806031, May 1999.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *ACM Symposium on the Theory of Computing (STOC) 1988*, pages 1–10, New York, NY, USA, 1988. ACM Press.

[BL96] Dan Boneh and Richard J. Lipton. Algorithms for black-box fields and their application to cryptography (extended abstract). In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 283–297. Springer-Verlag, 1996.

[BMM99] Amos Beimel, Tal Malkin, and Silvio Micali. The all-or-nothing nature of two-party secure computation. In Michael J. Wiener, editor, *Advances in Cryptology — CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 80–97. Springer, 1999.

[BPW04] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A general composition theorem for secure reactive systems. In Moni Naor, editor, *Theory of Cryptography Conference (TCC) 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 336–354. Springer, 2004.

[BS84] László Babai and Endre Szemerédi. On the complexity of matrix group problems I. In *IEEE Symposium on Symposium on Foundations of Computer Science (FOCS) 1984*, pages 229–240, Singer Island, Florida, 1984. IEEE.

[BS96] Eric Bach and Jeffrey Shallit. *Algorithmic Number Theory*, volume 1 of *Foundations of Computing*. MIT Press, Cambridge, Massachusetts, 1996.

[BT07] Anne Broadbent and Alain Tapp. Information-theoretic security without an honest majority. In *ASIACRYPT*, pages 410–426, 2007.

[BvzGL01] Eric Bach, Joachim von zur Gathen, and Hendrik W. Lenstra, Jr. Factoring polynomials over special finite fields. *Finite Fields and Their Applications*, 7:5–28, 2001.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In Bob Werner, editor, *IEEE Symposium on Symposium on Foundations of Computer Science (FOCS) 2001*, pages 136–147, Los Alamitos, CA, October 14–17 2001. IEEE Computer Society.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols. In *ACM Symposium on the Theory of Computing (STOC) 1988*, pages 11–19. ACM, 1988.

[CCM02] Christian Cachin, Claude Crépeau, and Julien Marcil. Oblivious transfer with a memory-bounded receiver. In *ACM Symposium on the Theory of Computing (STOC) 2002*, pages 493–502. ACM Press, 2002.

[CDD+99] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT 1999*, volume 1592

of *Lecture Notes in Computer Science*, pages 311–326. Springer-Verlag, May 1999.

[CDG88] David Chaum, Ivan Damgård, and Jeroen van de Graaf. Multiparty computations ensuring privacy of each party's input and correctness of the result. In *Advances in Cryptology — CRYPTO 1987*, pages 87–119, London, UK, 1988. Springer-Verlag.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *Advances in Cryptology — CRYPTO 2001*, pages 19–40, London, UK, 2001. Springer-Verlag.

[Cha89] David Chaum. The spymasters double-agent problem: Multiparty computation secure unconditionally from minorities and cryptographically from majorities. In G. Brassard, editor, *Advances in Cryptology — CRYPTO 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 591–602. Springer-Verlag, 1990, 20–24 August 1989.

[CK89] Benny Chor and Eyal Kushilevitz. A zero-one law for boolean privacy. In *ACM Symposium on the Theory of Computing (STOC) 1989*, 1989.

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty. In *ACM Symposium on the Theory of Computing (STOC) 1986*, pages 364–369, New York, NY, USA, 1986. ACM Press.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *ACM Symposium on the Theory of Computing (STOC) 2002*, pages 494–503, New York, NY, USA, 2002. ACM.

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

[DFSS05] Ivan Damgård, Serge Fehr, Louis Salvail, and Christian Schaffner. Cryptography in the bounded quantum-storage

model. In *IEEE Symposium on Symposium on Foundations of Computer Science (FOCS) 2005*, pages 449–458. IEEE Computer Society, 2005.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(5):644–654, 1976.

[DM04] Stefan Dziembowski and Ueli M. Maurer. On generating the initial key in the bounded-storage model. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology — EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 126–137. Springer, 2004.

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

[Dom02] Josep Domingo-Ferrer. A provably secure additive and multiplicative privacy homomorphism. In Agnes Hui Chan and Virgil D. Gligor, editors, *Information Security, 5th International Conference, ISC 2002*, volume 2433 of *Lecture Notes in Computer Science*, pages 471–483. Springer, 2002.

[DS83] Danny Dolev and Raymond H. Strong. Authenticated algorithms for byzantine agreement. *SICOMP: SIAM Journal on Computing*, 12, 1983.

[FHHW03] Matthias Fitzi, Martin Hirt, Thomas Holenstein, and Jürg Wullschleger. Two-threshold broadcast and detectable multiparty computation. In Eli Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, volume 265 of *Lecture Notes in Computer Science*, pages 51–67. Springer-Verlag, May 2003.

[FHW04] Matthias Fitzi, Thomas Holenstein, and Jürg Wullschleger. Multi-party computation with hybrid security. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology — EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 419–438. Springer-Verlag, May 2004.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *ACM Symposium on the Theory of Computing (STOC) 2009*, pages 169–178. ACM, 2009.

[GHKL08] S. Dov Gordon, Carmit Hazay, Jonathan Katz, and Yehuda Lindell. Complete fairness in secure two-party computation. In Richard E. Ladner and Cynthia Dwork, editors, *ACM Symposium on the Theory of Computing (STOC) 2008*, pages 413–422. ACM, 2008.

[GK08] Vipul Goyal and Jonathan Katz. Universally composable multi-party computation with an unreliable common reference string. In *Theory of Cryptography Conference (TCC) 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 142–154. Springer, 2008.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *ACM Symposium on the Theory of Computing (STOC) 1987*, pages 218–229, 1987.

[GO07] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In *Advances in Cryptology — CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 323–341. Springer, 2007.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, 2001.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, 2004.

[Her05] Amir Herzberg. On tolerant cryptographic constructions. In *CT-RSA'05*, volume 3376 of *Lecture Notes in Computer Science*, pages 172–190. Springer, 2005.

[HM00] Martin Hirt and Ueli Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, April 2000. Extended abstract in *Proc. 16th of ACM PODC '97*.

[HR07] Iftach Haitner and Omer Reingold. Statistically-hiding commitment from any one-way function. In David S. Johnson

and Uriel Feige, editors, *ACM Symposium on the Theory of Computing (STOC) 2007*, pages 1–10. ACM, 2007.

[IKLP06] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In C. Dwork, editor, *Advances in Cryptology — CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 483–500. Springer, 2006.

[Kat06] Jonathan Katz. On achieving the "best of both worlds" in secure multiparty computation. Cryptology ePrint Archive, Report 2006/455, 2006. `http://eprint.iacr.org/`.

[Kil91] Joe Kilian. A general completeness theorem for two-party games. In *ACM Symposium on the Theory of Computing (STOC) 1991*, pages 553–560, New York, 1991. ACM Press.

[Kil00] Joe Kilian. More general completeness theorems for secure two-party computation. In *ACM Symposium on the Theory of Computing (STOC) 2000*, pages 316–324, New York, 2000. ACM Press.

[KMQ08] Daniel Kraschewski and Jörn Müller-Quade. Completeness theorems with constructive proofs for symmetric, asymmetric and general 2-party-functions. Unpublished Manuscript, April 2008. Available at `http://iks.ira.uka.de/eiss/completeness`.

[KMQR09] Robin Künzler, Jörn Müller-Quade, and Dominik Raub. Secure computability of functions in the IT setting with dishonest majority and applications to long-term security. In Omer Reingold, editor, *Theory of Cryptography Conference (TCC) 2009*, volume 5444 of *Lecture Notes in Computer Science*, pages 238–255. Springer, 2009.

[Kus92] Eyal Kushilevitz. Privacy and communication complexity. *SIAM Journal on Discrete Mathematics*, 5(2):273–284, 1992.

[Lan02] Serge Lang. *Algebra*, volume 211 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 3rd edition, 2002.

[Len91] Hendrik W. Lenstra, Jr. Finding isomorphisms between finite fields. *Mathematics of Computation*, 56(193):329–347, 1991.

[Len05] Hendrik W. Lenstra, Jr. Personal Communication, 2005.

[LN97] Rudolf Lidl and Harald Niederreiter. *Finite Fields*, volume 20 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2nd edition, 1997.

[LRM09] Christoph Lucas, Dominik Raub, and Ueli Maurer. Optimally hybrid-secure MPC. IACR ePrint Archive, 2009. Available at `http://eprint.iacr.org/2009/009`.

[Mau] Ueli Maurer. Lecture notes cryptography.

[Mau94] Ueli Maurer. Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms. In Yvo Desmedt, editor, *Advances in Cryptology — CRYPTO 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 271–281. Springer-Verlag, 1994.

[Mau05] Ueli Maurer. Abstract models of computation in cryptography. In Nigel P. Smart, editor, *Cryptography and Coding 2005*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 2005.

[Mau09] Ueli Maurer. Abstraction in cryptography. In *Advances in Cryptology — CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, page 465. IACR, Springer, 2009. Slides available at `http://www.sti.uniurb.it/events/fosad09/slides/Ueli-FOSAD09.pdf`.

[MPR09] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Complexity of multi-party computation problems: The case of 2-party symmetric secure function evaluation. In Omer Reingold, editor, *Theory of Cryptography Conference (TCC) 2009*, volume 5444 of *Lecture Notes in Computer Science*, pages 256–273. Springer, 2009.

[MQ05] Jörn Müller-Quade. Temporary assumptions—quantum and classical. In *The 2005 IEEE Information Theory Workshop on Theory and Practice in Information-Theoretic Security*, pages 31–33, 2005.

[MQU07] J. Müller-Quade and D. Unruh. Long-term security and universal composability. In *Theory of Cryptography Conference*

(TCC) 2004, Lecture Notes in Computer Science. Springer-Verlag, 2007.

[MR07] Ueli Maurer and Dominik Raub. Black-box extension fields and the inexistence of field-homomorphic one-way permutations. In *Advances in Cryptology — ASIACRYPT 2007*, pages 427–443, 2007.

[MW99] Ueli Maurer and Stefan Wolf. The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms. *SIAM Journal on Computing*, 28(5):1689–1721, April 1999.

[PW00] Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *ACM Conference on Computer and Communications Security (ACM-CCS)*, pages 245–254, 2000.

[Rab03] Michael O. Rabin. Hyper-encryption by virtual satellite. Science Center Research Lecture Series, December 2003. Online available at `http://athome.harvard.edu/dh/hvs.html`.

[RB89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *ACM Symposium on the Theory of Computing (STOC) 1989*, pages 73–85, New York, NY, USA, 1989. ACM Press.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–268. Springer-Verlag, 1997.

[SYY99] Tomas Sander, Adam Young, and Moti Yung. Non-interactive CryptoComputing for NC^1. In *IEEE Symposium on Symposium on Foundations of Computer Science (FOCS) 1999*, pages 554–567, New York, NY, USA, 1999. IEEE Computer Society Press.

[Wul07] Jürg Wullschleger. *Oblivious-Transfer Amplification*. PhD thesis, ETH Zürich, 2007.

[Yao82] Andrew C. Yao. Protocols for secure computations (extended abstract). In *IEEE Symposium on Symposium on Foundations of*

Computer Science (FOCS) 1982, pages 160–164, Chicago, Illinois, 1982. IEEE.

Appendix A

Odds and Ends

A.1 Perfectly Hiding or Perfectly Binding UC Commitments

We describe a UC secure one-to-many commitment schemes implementing the one-to-many commitment functionality $\mathsf{F}_{\mathsf{Com},1:\mathsf{M}}$ that can be either perfectly hiding or perfectly binding. Our one-to-many commitment scheme is derived from the perfectly hiding or perfectly binding UC secure commitment schemes of [DN02].

Functionality $\mathsf{F}_{\mathsf{ComH},1:\mathsf{M}}$ formalizes perfectly hiding UC secure one-to-many commitment schemes, functionality $\mathsf{F}_{\mathsf{ComB},1:\mathsf{M}}$ formalizes perfectly binding UC secure one-to-many commitment schemes.

Functionality $\mathsf{F}_{\mathsf{ComH},1:\mathsf{M}}$ operates as follows:

1. If the committer C is honest or in the CO setting:
   (a) On receipt of a message m from the committer C, output committed to all receivers R_i.
   (b) On receipt of open from the committer C, output m to all receivers R_i.
2. If the committer C is corrupted in the IT setting, turn control over to the adversary.

Functionality $\mathsf{F}_{\mathsf{ComB},1:\mathsf{M}}$ operates as follows:

1. For honest receivers R_i or in the CO setting for all receivers R_i:
   (a) On receipt of a message m from the committer C, output committed to the receiver R_i.
   (b) On receipt of open from the committer C, output m to the receiver R_i.
2. For corrupted receivers R_i in the IT setting,
   (a) On receipt of a message m from the committer C, directly output m to the receiver R_i.
   (b) On receipt of open from the committer C, output open to the receiver R_i.

We no show how to extend the commitment scheme of [DN02] to implement the functionalities $\mathsf{F}_{\mathsf{ComH},1:\mathsf{M}}$ and $\mathsf{F}_{\mathsf{ComB},1:\mathsf{M}}$.

A.1.1 Mixed Commitments

The construction of our UC commitment scheme is based on the mixed commitment scheme commit_K described in [DN02]. A mixed commitment scheme is parameterized by a system key N with an associated X-trapdoor t_N which determines keyspace $\mathcal{K}_N$ and message space $\mathcal{M}_N$. Both $\mathcal{K}_N$ and $\mathcal{M}_N$ are additive groups. The keyspace $\mathcal{K}_N$ is partitioned into subsets $\mathcal{K}_X$ of X-keys (for extractability), $\mathcal{K}_E$ of E-keys (for equivocability), and $\mathcal{K}_R$ of remaining keys. An overwhelming fraction of keys in $\mathcal{K}_N$ are X-keys in $\mathcal{K}_X$. One can efficiently generate random system keys N, random keys in $\mathcal{K}_N$, random X-keys in $\mathcal{K}_X$, and random E-keys in $\mathcal{K}_E$. All X-keys $K \in \mathcal{K}_X$ have a common trapdoor t_N that can efficiently be generated together with the system key N. In contrast, all E-keys $K \in \mathcal{K}_E$ have their own trapdoor t_K that can efficiently be generated together with the key itself. Furthermore, random keys, X-keys, and E-keys are CO indistinguishable.

The commitment scheme commit_K with key $K \in \mathcal{K}_N$ is equivocable for $K \in \mathcal{K}_E$ and extractable for $K \in \mathcal{K}_X$. So, on the one hand, for a commitment $c = \mathsf{commit}_K(m, r)$ where $K \in \mathcal{K}_X$ one can efficiently determine m from c, K, N, and the trapdoor t_N (extractability). On the other hand, given $K \in \mathcal{K}_E$ and the associated E-trapdoor t_K one can efficiently generate a commitment c that is equivocable, i.e. it is efficiently possible to generate randomness r such that $c = \mathsf{commit}_K(m, r)$ for any $m \in \mathcal{M}_N$.

Note that extractability and equivocability together with the CO indistinguishability of random keys, X-keys, and E-keys imply that the mixed commitment scheme commit_K is CO binding and hiding. More details on mixed commitments can be found in [DN02].

A.1.2 The CRS

UC commitments require a stronger setup than a broadcast channel [CF01]. We will use a common random string (CRS) that is sampled from a prescribed distribution by a trusted functionality.

Our CRS will be $\mathsf{CRS} = (N, K_X, K_E, \bar{K}_1, \ldots, \bar{K}_n, \mathsf{CRS}')$. The first part of the CRS, encompassing the $n+3$ keys $N, K_X, K_E, \bar{K}_1, \ldots, \bar{K}_n$, stems from the original protocol in [DN02]. In accordance to this protocol, N is a random system key for our mixed commitment, $K_X \in \mathcal{K}_X$ is a random X-key and $K_E, \bar{K}_1, \ldots, \bar{K}_n \in \mathcal{K}_E$ are random E-keys. The second part of the CRS, i.e. CRS', is a CRS for one-to-many commitments according to [CLOS02, CF01]. This part is only needed for the one-to-many extension of the commitment scheme discussed here.

A.1.3 The UC Commitment Protocol

Wlog let $C = P_1$ be the committer and the remaining parties be the receivers $R_i = P_i$ ($i \in 2, \ldots, n$). Furthermore, let $(\mathsf{commit}', \mathsf{open}')$ denote the one-to-many commitment scheme according to [CLOS02, CF01]. The UC one-to-many commitment protocol then works as follows:

Commit phase:

C.1 On input m, committer C draws a random $K_1 \in \mathcal{K}_N$ and random opening information r_1, and broadcasts $c_1 = \mathsf{commit}_{\bar{K}_1}(K_1, r_1)$.

R.1 The receivers R_i run a coin toss protocol in order to sample a random key K_2:

- **R.1.a** Each receiver R_i draws a random $s_i \in \mathcal{K}_N$, computes $(c'_i, o'_i) = \mathsf{commit}'(s_i, \mathsf{CRS}')$, and broadcasts c'_i.
- **R.1.b** Each receiver R_i broadcasts (s_i, o_i).
- **R.1.c** All parties compute $K_2 = \sum_i s_i$ for the s_i where $s_i = \mathsf{open}'(c'_i, o'_i)$.

C.2 Committer C computes $K = K_1 + K_2$, draws random opening information r_2, r_3, and

- for an IT hiding commitment draws $\bar{m}$ and broadcasts the commitments $c_2 = \mathsf{commit}_K(\bar{m}+m, r_2)$, $c_3 = \mathsf{commit}_{K_E}(\bar{m}, r_3)$
- for an IT binding commitment broadcasts the commitments $c_2 = \mathsf{commit}_K(m, r_2)$, $c_3 = \mathsf{commit}_{K_X}(m, r_3)$

R.2 Each receiver R_i upon receiving c_2 and c_3 outputs committed

Opening phase:

C.1 On input open, committer C broadcasts

- for an IT hiding commitment $(m, \bar{m}, r_2, r_3)$
- for an IT binding commitment (m, r_2, r_3)

R.1 Each receiver R_i verifies that

- for an IT hiding commitment $c_2 = \mathsf{commit}_K(\bar{m} + m, r_2)$, $c_3 = \mathsf{commit}_{K_E}(\bar{m}, r_3)$,
- for an IT binding commitment $c_2 = \mathsf{commit}_K(m, r_2)$, $c_3 = \mathsf{commit}_{K_X}(m, r_3)$

and if so, outputs m.

Note that this protocol is a simple adaption of [DN02] to multiple receivers. We simply replace round R.1 of [DN02] where the single receiver of [DN02] chooses a random K_2 with a CO secure cointoss among our multiple receivers.

A.1.4 Security of the UC Commitment Protocol

We prove security by providing simulators for the IT hiding and the IT binding case separately. The argument why these simulators achieve indistinguishability does not change substantially and we refer the reader to [DN02].

A.1.4.1 IT Hiding

We now show that the perfectly hiding variation of the scheme above indeed implements functionality $\mathsf{F}_{\mathsf{ComH,1:M}}$. We consider three cases for which we provide different simulators, namely:

1. the adversary is CO or IT, leaves the committer C honest (and corrupts any number of receivers R_i).
2. the adversary is CO, corrupts the committer C (and any number of receivers R_i),
3. the adversary is IT, corrupts the committer C (and any number of receivers R_i).

In the first two cases the commitment functionality $\mathsf{F}_{\mathsf{ComH,1:M}}$ operates as expected and described in [CF01]. Simulator $\mathsf{S}_R^{\mathsf{it}}$ is used in case 1 where C is honest, but any number of receivers R_i are IT or CO corrupted (the simulator works in both cases). First, $\mathsf{S}_R^{\mathsf{it}}$ produces a regular CRS with E-key K_E and E-trapdoor t_E. During the commit phase, $\mathsf{S}_R^{\mathsf{it}}$ emulates C on random input to the corrupted R_i. Indistinguishability is preserved because all commitments are equivocable and thus independent of their "content". In the opening phase, $\mathsf{S}_R^{\mathsf{it}}$ receives the correct $m^\star$ from $\mathsf{F}_{\mathsf{com}}^h$. Now, $\mathsf{S}_R^{\mathsf{it}}$ opens the K_E commitment c_3 to $m_3' = m^\star \oplus m_2$ using t_E.

Finally, simulator $\mathsf{S}_C^{\mathsf{co}}$ is used in case 2 where C and any number of receivers are CO corrupted. First, $\mathsf{S}_C^{\mathsf{co}}$ produces a fake $\widetilde{\mathsf{CRS}}$ with interchanged keys: On one hand, in $\widetilde{\mathsf{CRS}}$, K_X is an *equivocable* key taken from $\mathcal{K}_E$, together with trapdoor t_{K_X} for equivocability. On the other hand, in $\widetilde{\mathsf{CRS}}$, K_E is an *extractable* key taken from $\mathcal{K}_X$. Note that K_E has trapdoor t_N for extractability. For a CO adversary, the fake $\widetilde{\mathsf{CRS}}$ is indistinguishable from a real CRS. Furthermore, $\mathsf{S}_C^{\mathsf{co}}$ internally runs the protocol of honest R_i which can be perfectly simulated since they do not require any input. During the simulation, $\mathsf{S}_C^{\mathsf{co}}$ simply forwards all messages among the (internally simulated) honest R_i and the corrupted parties, i.e. C and corrupted R_i. After the commit phase, $\mathsf{S}_C^{\mathsf{co}}$ uses the known system trapdoor N to extract m_3 from c_3 (X-key by choice of the CRS) and m_2 for c_2 (X-key with overwhelming probability in the regular protocol) and inputs $m^\star = m_3 \oplus m_2$ to $\mathsf{F}_{\mathsf{com}}^h$. In the opening phase, $\mathsf{S}_C^{\mathsf{co}}$ sends an open message to $\mathsf{F}_{\mathsf{com}}^h$ if and only if C provides correct opening information for $m^\star$.

In the last case, committer C and any number of receivers are IT corrupted. By definition of IT hiding commitments, the functionality $\mathsf{F}_{\mathsf{com}}^{h}$ collapses in this context and turns over control to the simulator $\mathsf{S}_C^{\mathsf{it}}$. Our simulator $\mathsf{S}_C^{\mathsf{it}}$ first produces a regular CRS. Then, $\mathsf{S}_C^{\mathsf{it}}$ internally runs the protocol of the honest R_i to the I/O-interface of which it has access via the ideal functionality. During the simulation, $\mathsf{S}_C^{\mathsf{it}}$ simply forwards all messages among the (internally simulated) honest R_i and the corrupted parties, i.e. C and corrupted R_i.

As noted above, the indistinguishability arguments of [DN02] apply, and we refer the reader there for further detail.

A.1.4.2 IT Binding

We now show that the perfectly binding variation of the scheme above indeed implements functionality $\mathsf{F}_{\mathsf{ComB,1:M}}$. Once again, we consider three cases for which we provide different simulators, namely:

1. the adversary is CO or IT, corrupts the committer C and any number of receivers R_i,
2. the adversary is CO, leaves the committer C honest and corrupts any number of receivers R_i,
3. the adversary is IT, leaves the committer C honest and corrupts any number of receivers R_i.

In the first two cases the commitment functionality operates as expected and described in [CF01]. Simulator $\mathsf{S}_C^{\mathsf{it}}$ is used in case 1 where C and any number of receivers are IT or CO corrupted. First, $\mathsf{S}_C^{\mathsf{it}}$ produces a regular CRS with X-key K_X and X-trapdoor t_N. Furthermore, $\mathsf{S}_C^{\mathsf{it}}$ internally runs the protocol of honest R_i which can be perfectly simulated since they do not require any input. During the simulation, $\mathsf{S}_C^{\mathsf{it}}$ simply forwards all messages among the (internally simulated) honest R_i and the corrupted parties, i.e. C and corrupted R_i. After the commit phase, $\mathsf{S}_C^{\mathsf{it}}$ extracts the message m from c_3 using the trapdoor t_N, and enters it into $\mathsf{F}_{\mathsf{com}}^{b}$. In the opening phase, $\mathsf{S}_C^{\mathsf{it}}$ sends an open message to $\mathsf{F}_{\mathsf{com}}^{b}$ if and only if C provides correct opening information for m.

Simulator $\mathsf{S}_R^{\mathsf{co}}$ is used in case 2 where C is honest, but any number of receivers R_i is CO corrupted. First, $\mathsf{S}_R^{\mathsf{co}}$ produces a fake $\widetilde{\mathsf{CRS}}$ with interchanged keys: On one hand, in $\widetilde{\mathsf{CRS}}$, K_X is an *equivocable* key taken

from $\mathcal{K}_E$, together with trapdoor t_{K_X} for equivocability. On the other hand, in $\widetilde{\mathsf{CRS}}$, K_E is an *extractable* key taken from $\mathcal{K}_X$. Note that K_E has trapdoor t_N for extractability. For a CO adversary, the fake $\widetilde{\mathsf{CRS}}$ is indistinguishable from a real CRS. In the commit phase in step C.1, $\mathsf{S}_R^{\mathsf{co}}$ uses the trapdoor $t_{\bar{K}_C}$ of E-key $\bar{K}_C$ to produce an equivocable commitment c_1. Hence, C is not committed to the first part K_1 of the key K. Then, in step C.2, $\mathsf{S}_R^{\mathsf{co}}$ opens c_1 to a value K_1' such that $K = K_1' \oplus K_2$ is a random E-key with known trapdoor t_{K_E}. Usually, this would be an X-key with overwhelming probability. For the opening phase, $\mathsf{S}_R^{\mathsf{co}}$ receives the correct m from $\mathsf{F}_{\mathsf{com}}^h$. Then, $\mathsf{S}_R^{\mathsf{co}}$ opens the commitment c_3 and the commitment c_2 to $m_3' = m_2' = m$. By choice of the $\widetilde{\mathsf{CRS}}$, the commitment c_3 was constructed with the equivocable E-key K_X and trapdoor t_{K_X}. By choice of K_1', the commitment c_2 was constructed with the equivocable E-key $K = K_1' \oplus K_2$ and trapdoor t_K. Hence, $\mathsf{S}_R^{\mathsf{co}}$ can efficiently open both commitments as needed.

In the last case, committer C is honest, but any number of receivers R_i is IT corrupted. By definition of IT binding commitments, the ideal functionality $\mathsf{F}_{\mathsf{com}}^b$ then directly leaks the committed message m to IT corrupted receivers. Honest R_i still receive m from $\mathsf{F}_{\mathsf{com}}^b$ on opening as usual. We use a simulator $\mathsf{S}_R^{\mathsf{it}}$ that exploits this. First, $\mathsf{S}_R^{\mathsf{it}}$ produces a regular CRS. Then, it internally runs the protocol of C on input m, and the protocols of honest R_i, which do not need any input, towards the corrupted R_i.

As noted above, the indistinguishability arguments of [DN02] apply, and we refer the reader there for further detail.

Curriculum Vitae

Dominik Léon Raub,
German citizen,
born 1979/04/15 in Freudenstadt, Germany,
as son of Wolfgang and Mona Raub.

Secondary School

School: Tulla Gymnasium Rastatt, Germany. 1989–1998.
Diploma: Abitur.

Civil Service

Employer: Caritas Rastatt. 1998–1999.

Fulbright Scholarship

University: Carnegie Mellon University. 2002–2003.

Diploma Studies

University: Universität Karlsruhe, Fakulät für Informatik. 1999–2005.
Thesis: "Algebraische Spezifikation von Privacy Policies".
Diploma: Degree in Computer Science (Dipl.-Inform.).

Doctoral Studies

University: ETH Zurich, Dept. of Computer Science. 2005–2009.
Thesis: "Exploring the Limits of Multi-Party Computation".
Examiner and Advisor: Prof. Dr. Ueli Maurer.
Co-examiner: Prof. Dr. Ronald Cramer, CWI and University of Leiden, The Netherlands.