Combining Computational and Information-Theoretic Security in Multi-Party Computation

Christoph Lucas, Dominik Raub, and Ueli Maurer

Department of Computer Science, ETH Zurich, Switzerland,
{clucas, raubd, maurer}@inf.ethz.ch

Abstract. Most protocols for multi-party computation (MPC) are secure either against information-theoretic (IT) or against computationally bounded adversaries. In this work, we bring together the best of both worlds: For any robustness parameter $\rho < \frac{n}{2}$ we obtain one MPC protocol that is simultaneously IT secure with robustness for up to $t \leq \rho$ actively corrupted parties, IT secure with fairness (no robustness) for up to $t < \frac{n}{2}$ and computationally secure with agreement on abort (no fairness) for up to $t < n - \rho$. Our construction is secure in the universal composability (UC) framework, and achieves the bounds of Ishai et al. [CRYPTO'06], Katz [STOC'07], and Cleve [STOC'86] on trade-offs between robustness and privacy, and on fairness.
For example, for the special case $\rho = 0$ our protocol simultaneously achieves non-robust MPC for up to $t < n$ corrupted parties in the computational setting (like Goldreich et al. [STOC'87]) while providing security with fairness in the IT setting for up to $t < \frac{n}{2}$ corrupted parties (like Rabin and Ben-Or [STOC'89] though without robustness).

Keywords: multi-party computation, information-theoretic security, computational security, hybrid security, robustness, fairness, agreement on abort, universal composability.

1 Introduction

1.1 Secure Multi-Party Computation

In [21], Yao introduced multi-party computation (MPC): Given any arbitrary but fixed function f and a set of n mutually distrusting parties, an MPC protocol enables these parties to compute the function f on their inputs securely, even if some of the parties are corrupted by an active adversary.

Security notions for MPC in the literature (e.g. [13]) are numerous and include privacy, correctness, robustness, fairness, and agreement on abort. *Privacy* is achieved if the adversary cannot learn more about the honest players' inputs than what is implied by inputs and outputs of the corrupted players. *Correctness* means that the protocol output equals the intended function value $f(x_1, ..., x_n)$ of the inputs or there is no output. A protocol achieves *robustness* if an adversary cannot abort it and prevent the honest players from obtaining output. *Fairness* is achieved if the honest parties learn the output whenever the adversary learns anything about it. *Agreement on abort* means that either all honest parties obtain the output or none. *Full security* encompasses all of the above.

A first general solution to the MPC problem was given by [15], based on computational (CO) intractability assumptions and a broadcast (BC) channel. They achieve full security against $t < \frac{n}{2}$ actively corrupted parties or security with abort against $t < n$ actively corrupted parties as described in [14]. On the other hand [2], and independently [7], 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) [11] or if a BC channel is available [19], this bound can be extended to at most $t < \frac{n}{2}$.

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) [3, 4]. Unless otherwise stated we assume a complete and synchronous network of secure channels, an authenticated synchronous BC channel, and a CRS.

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.

The goal of MPC protocols with hybrid security is to provide different levels of security, depending on the number of corrupted parties. For instance we will discuss protocols that are IT secure with fairness in the presence of $t < \frac{n}{2}$ actively corrupted parties and still CO secure with agreement on abort in the presence of arbitrarily many corrupted parties.

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 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 $t < n - \rho$.

In [6] Chaum sketches a construction aimed at guaranteeing CO security with abort for any number of actively corrupted parties, and simultaneously IT security with abort, given that only a minority of the parties is corrupted. However, several critical details are neglected in [6]. We provide a protocol construction based on Chaum's idea and give a formal and complete description. Our construction improves upon [6] by guaranteeing fairness in the IT setting, agreement on abort in the CO setting and a method for trading robustness against at most ρ corrupted parties for CO security against up to $n - \rho$

corrupted parties. Furthermore, we provide proofs in the UC model as well as the stand-alone model (see full version).

Fitzi et al. [12] combine IT and CO security: Up to a first threshold t_p, the security is unconditional. Between t_p and the second threshold t_σ IT security is guaranteed conditional 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.

Fitzi et al. [11] improve upon [2, 7] 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 [12] and [11] largely focus on a setting without BC channel. When a BC channel is provided our results improve substantially upon those of [12, 11]. As [11] focuses exclusively on IT MPC and [12] only treats robust MPC both [12, 11] 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 [11] (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 [11].

In [16] and [17] 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 [16, 17] but beyond that also provides IT security for $t < \frac{n}{2}$. Futhermore, we match the bound $t < \frac{n}{2}$ on fairness for general MPC put forth in [9], and the bound $t < \frac{n}{2}$ on IT security in the presence of active adversaries (e.g. [18]).

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.

2 Security Definitions and Notations

We use the universal composability (UC) model of security [3, 1][1] which is a simulation-based security model. The security of a protocol (the real model) is defined with respect to an ideal model, where the computation is performed by a *Trusted Third Party* or *Ideal Functionality* $\mathcal{I}$. A protocol π is to implement an ideal functionality $\mathcal{I}$ from a given set of resources (e.g. authentic or secure channels, BC channels, a PKI) using local protocol machines π_i for each party P_i. Informally, a protocol π achieves security according to the simulation paradigm if whatever an adversary A corrupting a subset $\mathfrak{A} \subseteq [n]$ (where $[n] = \{1, 2, ..., n\}$)[2] of $t := |\mathfrak{A}|$ parties can do in the real model, a simulator

[1] We follow the UC model of [3] in spirit, but do not adhere to the notation of [3].

[2] We will occasionally refer to party P_i by its index i, and e.g. write $P_i \in \mathfrak{A}$.

(or ideal adversary) σ could also do in the ideal model. This is formalized by means of a distinguisher (or environment) D, which connects to the protocol interfaces of the honest parties $P_i \in \mathfrak{H} = [n] \setminus \mathfrak{A}$ and to the adversary. A protocol π is secure if there is a simulator σ such that no distinguisher D can tell real model and ideal model with simulator σ apart.

As (in contrast to stand-alone definitions of security) the distinguisher is not restricted to providing input and receiving output, but may interact with the adversary or simulator during the entire course of the protocol execution, we implicitly model that other protocols may be running in the environment (i.e. the distinguisher) in parallel and that the adversary may swap messages or otherwise use information from one protocol in another. This yields a strong composition theorem [3, 1] which allows us to securely replace a protocol component with the ideal functionality that it securely implements in any protocol context.

As shown in [3] it is not necessary to consider a proper adversary A. Rather the (real model) adversary can be thought of as part of the distinguisher D (dummy adversary in [3]). D then directly connects to the resource interfaces of the corrupted parties. Security is exhibited by providing a simulator acting on the interfaces of the corrupted parties in the ideal setting that renders the ideal functionality with simulator and the real protocol indistinguishable for any such distinguisher. Indistinguishability is formalized by means of the advantage that a distinguisher has in distinguishing two system:

Definition 1 (Advantage of a Distinguisher). *The advantage of a class $\mathcal{D}$ of distinguishers in distinguishing systems S and S' is* $\mathrm{adv}^{\mathcal{D}}(S, S') := \max_{D \in \mathcal{D}} |\mathrm{Pr}[D(S) = 1] - \mathrm{Pr}[D(S') = 1]|$, *where $D(S)$ denotes the output of distinguisher D when interacting with system S.*

UC security then essentially states that wherever a protocol π is used, we can indistinguishably replace this protocol by the ideal functionality $\mathcal{I}$ it implements together with an appropriate simulator:

Definition 2 (Universally Composable (UC) Security). *Given a class of distinguishers $\mathcal{D}$ and a class of simulators Σ, a protocol π UC-securely implements an ideal functionality $\mathcal{I}$ if for every set of corrupted players $\mathfrak{A}$ there is a simulator $\sigma \in \Sigma$ such that* $\mathrm{adv}^{\mathcal{D}}(\sigma_{\mathfrak{A}}(\mathcal{I}), \pi_{\mathfrak{H}}(R))) \leq \epsilon(\kappa)$, *where $\epsilon(\kappa)$ is a negligible function in the security parameter κ. Here $\pi_{\mathfrak{H}}$ denotes the protocol machines of the honest parties in $\mathfrak{H}$. We generally take Σ as the class of* efficient *simulators. If $\mathcal{D}$ is the class of* efficient *distinguishers we obtain CO security, if $\mathcal{D}$ is the class of* all *distinguishers we obtain IT security.*

We will generally be interested in securely computing an n party function f that takes n inputs $x_1, \ldots, x_n$, one from each party, and maps them to a common output $y = f(x_1, \ldots, x_n)$. At first glance it might appear more natural to consider asymmetric functions which provide a separate output for each party: $(y_1, ..., y_n) = f(x_1, ..., x_n)$. However, a simple argument shows that asymmetric MPC can IT securely be reduced to symmetric MPC (see e.g. [8], there called *private output*). Thus we may restrict our attention to symmetric functions. In some cases this work still uses indexed outputs y_i. However, this is not an indication of asymmetric output, but denotes the output received at the interface of party P_i. As is common for MPC, we assume that the function f is given as a circuit over a finite field $\mathbb{F}$. All computations are then performed over the finite field $\mathbb{F}$.

We model the computation of a function f with a specific set of security properties by providing an ideal functionality for each such set.

Demanding *privacy, correctness and agreement on abort* only for the computation of f is captured by the ideal functionality $\mathcal{I}_f^{abort}$, which operates as follows: $\mathcal{I}_f^{abort}$ accepts an input from each party, computes the output y according to f and sends y to the adversary (simulator). If a party P_i provides no input, a default input is used, e.g. $x_i = 0$. Given these values, the adversary may decide whether the other parties also receive the output (abort flag 0) or not (abort flag 1). Finally, $\mathcal{I}_f^{abort}$ sends either the output y or the empty value $\perp$ to the honest parties, depending on the flag received from the adversary.[3]

The ideal functionality $\mathcal{I}_f^{fair}$ specifying *privacy, correctness and fairness* (which implies agreement on abort) works like $\mathcal{I}_f^{abort}$ but takes an abort flag *before* making output to the adversary. Then for abort flag 0 the functionality $\mathcal{I}_f^{abort}$ sends the result y to *all* parties and for abort flag 1 it sends $\perp$ to *all* parties.

Computing function f with *full security*, which implies all the security notions mentioned above, is specified by means of the ideal functionality $\mathcal{I}_f^{sec}$. The functionality $\mathcal{I}_f^{sec}$ operates like $\mathcal{I}_f^{fair}$ but takes no abort flag and instead directly delivers the output y to *all* parties.

3 The Protocol Π^ρ

We present a protocol Π^ρ that, for any choice of robustness parameter $\rho < \frac{n}{2}$, simultaneously provides IT security with robustness in the presence of up to $t \leq \rho$ actively corrupted parties, IT security with fairness (no robustness) for $t < \frac{n}{2}$ and CO security with agreement on abort (no fairness) for $t < n - \rho$. Formally stated:

Theorem 1 (UC Security of Π^ρ). *For any $\rho < \frac{n}{2}$, Π^ρ implements*

1. *the ideal model $\mathcal{I}_f^{sec}$ (with robustness), with IT security, given that $t \leq \rho$,*
2. *the ideal model $\mathcal{I}_f^{fair}$ (with fairness), with IT security, given that $t < \frac{n}{2}$, and*
3. *the ideal model $\mathcal{I}_f^{abort}$ (with agreement on abort), with CO security, given that $t < n - \rho$,*

from a complete and synchronous network of secure channels, an authenticated BC channel, and a CRS-setup, in the UC setting, where t is the number of (actively) corrupted parties.

Our construction uses the idea of emulating a player sketched in [6]. We employ a CO secure MPC protocol $\pi^{\mathrm{co},\rho}$ executed among $n+1$ parties $P_1,\ldots,P_n$, and P_0, which

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

is tweaked to provide additional IT security guarantees for the designated party P_0. Now, in contrast to the parties P_1,...,P_n, we make P_0 a virtual party, emulated by an IT secure protocol π^{it} run among the parties P_1,...,P_n.

As a consequence, each party P_i runs two protocol machines $\pi_i^{\text{co},\rho}$ and π_i^{it}, participating both in the CO protocol $\pi^{\text{co},\rho}$ and in the IT protocol π^{it} where π^{it} emulates the protocol machine $\pi_0^{\text{co},\rho}$ of the designated $(n+1)$st party P_0 of subprotocol $\pi^{\text{co},\rho}$. Party P_i locally shares its input into $x_i = x_i^{\text{co}} \oplus x_i^{\text{it}}$ submitting x_i^{it} to π_i^{it} and x_i^{co} to $\pi_i^{\text{co},\rho}$. The CO secure subprotocol $\pi^{\text{co},\rho}$ then essentially computes the output $y = f'((x_1^{\text{it}},...,x_n^{\text{it}}), x_1^{\text{co}},...,x_n^{\text{co}}) := f(x_1^{\text{co}} \oplus x_1^{\text{it}},...,x_n^{\text{co}} \oplus x_n^{\text{it}}) = f(x_1,...,x_n)$, where $(x_1^{\text{it}},...,x_n^{\text{it}})$ is the input of P_0.

We now describe the protocol machine π_i for party P_i in protocol Π^ρ in detail[4] (also see Figures 1,2):

Initial Situation: Each party P_i $i \in [n]$ holds an input x_i.
Goal: Each party P_i receives the output $y = f(x_1,...,x_n)$ or the empty value $\perp$.

1. Choose x_i^{it} uniformly at random and compute $x_i^{\text{co}} := x_i \oplus x_i^{\text{it}}$ (over $\mathbb{F}$).
2. Using an IT hiding commitment scheme compute $[c_i, o_i] = \text{COMMIT}(x_i^{\text{it}})$.
3. Start protocol machine π_i^{it} on input (x_i^{it}, o_i) emulating $\pi_0^{\text{co},\rho}$ IT securely as stated in Lemma 2 below.
4. Start protocol machine $\pi_i^{\text{co},\rho}$ for f according to Lemma 3 below on input (x_i^{co}, c_i).
5. Start a protocol machine π_i^v implementing an $n+1$ party BC (see Section 3.1).
6. Establish channels between (a) π_i^v and the BC interface of $\pi_i^{\text{co},\rho}$, (b) π_i^v and the BC channel BC, (c) π_i^v and π_i^{it}, and (d) π_i^v and the interface of $\pi_i^{\text{co},\rho}$ for the private channel with $\pi_0^{\text{co},\rho}$.
7. If a callback to P_i occurs from $\pi_i^{\text{co},\rho}$, P_i provides input x_i^{it} to $\pi_i^{\text{co},\rho}$.

By sharing the input between $\pi^{\text{co},\rho}$ and π^{it} we achieve IT security as long as the correct functioning of the emulated party P_0 (i.e. the IT protocol π^{it}) is guaranteed, and CO security beyond. The commiments and callbacks are necessary to ensure correctness in case $\frac{n}{2} \leq t < n - \rho$ parties are corrupted. Then π^{it} provides no security guarantees, so the emulated party P_0 may misbehave arbitrarily and must hence be CO tied to the correct inputs.

We now describe the components of the above protocol Π^ρ in detail, beginning with the IT secure subprotocol π^{it} emulating the designated party P_0.

3.1 Emulating the Designated Party P_0: The IT secure Protocol π^{it}

For the emulation of the protocol machine $\pi_0^{\text{co},\rho}$ of the designated party P_0 in subprotocol $\pi^{\text{co},\rho}$ we use an IT secure MPC subprotocol π^{it} as guaranteed by the following lemma taken from [19, 3]:

[4] We describe the two subprotocols $\pi^{\text{co},\rho}$ and π^{it} as relying on (separate) BC channels. However, we assume only a single BC channel for the entire protocol. This is no contradiction as the reader may verify that the BC channel can easily be "multiplexed".

Lemma 2 ([19, 3]) *Given a (well-formed, see [3]) ideal functionality $\mathcal{I}$ there is a protocol π^{it} that (robustly) implements the ideal model $\mathcal{I}$ with IT security, from a complete and synchronous network of secure channels, and a BC channel, in the UC setting, given that $t < \frac{n}{2}$ parties are corrupted.*

After receiving input (x_i^{it}, o_i) from P_i $(i \in [n])$, protocol machine $\pi_0^{\text{co},\rho}$ expects to interact with the $\pi_i^{\text{co},\rho}$ $(i \in [n])$ via private channels and an $n+1$ party BC channel BC'. However, only interfaces to the n parties P_i executing protocol π^{it} are available. We solve this dilema by defining an n party ideal functionality $\mathcal{I}_0^{sec}$ that specifies the protocol π^{it} and protocol machines[5] π_i^v $(i \in [n])$ that together, when applied to the n party BC channel BC (available by assumption), securely implement the protocol machine $\pi_0^{\text{co},\rho}$ and an $n+1$ party BC channel BC'.

Protocol machine π_i^v connects to $\mathcal{I}_0^{sec}$ on the interface of P_i, to BC on the interface of P_i and to $\pi_i^{\text{co},\rho}$ on the interfaces to BC^{n+1} and to $\pi_0^{\text{co},\rho}$. π_i^v then forwards private messages between $\mathcal{I}_0^{sec}$ and $\pi_i^{\text{co},\rho}$. BC messages from $\mathcal{I}_0^{sec}$ are marked as originating from $\pi_0^{\text{co},\rho}$ and tranfered to BC. If in turn at least $\frac{n}{2}$ such marked messages m_j from parties P_j are received via BC and there is a majority $\exists m : |\{m_j | m_j = m\}| > \frac{n}{2}$ then π_i^v forwards m to the BC interface of $\pi_i^{\text{co},\rho}$. BC messages from $\pi_i^{\text{co},\rho}$ are forwarded to BC and copied to $\mathcal{I}_0^{sec}$. BC messages from $\pi_j^{\text{co},\rho}$ $(i \neq j)$ forwarded to the BC interface of $\pi_i^{\text{co},\rho}$ and copied to $\mathcal{I}_0^{sec}$.

The n party ideal functionality $\mathcal{I}_0^{sec}$ internally runs the protocol machine $\pi_0^{\text{co},\rho}$ on the input it receives over the interfaces P_i, $i \in [n]$. Private communication intended for a protocol machine $\pi_i^{\text{co},\rho}$ is exchanged via the interface of party P_i. Messages to be broadcasted by $\pi_0^{\text{co},\rho}$ are instead sent to all parties P_i. A BC message m from one of the P_i $(i \in [n])$ is forwarded (by the functionality $\mathcal{I}_0^{sec}$) to the internally run $\pi_0^{\text{co},\rho}$ only if $\mathcal{I}_0^{sec}$ receives the BC message m as input from more than $\frac{n}{2}$ parties P_i.

This construction exploits the fact that π_i^v and π_i^{it} are corrupted if and only if $\pi_i^{\text{co},\rho}$ is corrupted as all three protocols are run as subprotocols of π_i by the same party P_i. Therefore if $\pi_i^{\text{co},\rho}$ is uncorrupted, then $\pi_i^{\text{co},\rho}$ and $\pi_0^{\text{co},\rho}$ share a secure channel by virtue of π_i^v providing a secure connection. If on the other hand $\pi_i^{\text{co},\rho}$ is corrupted, we need not worry about the security of this channel. Furthermore, the construction implements an $n+1$ party BC channel BC'. If at most $t < \frac{n}{2}$ parties are corrupted and subprotocol π^{it} is thus secure, the honest π_i^v (more than $\frac{n}{2}$) are all forwarding broadcast messages of protocol $\pi^{\text{co},\rho}$ to and from $\mathcal{I}_0^{sec}$ which implies persistency. It now suffices to see that persistency for senders other than $\pi_0^{\text{co},\rho}$ and consistency is maintained even for $t \geq \frac{n}{2}$, as for $t \geq \frac{n}{2}$ protocol π^{it} is insecure and we regard party P_0 as corrupted. This holds due to the fact that BC messages from senders other than $\pi_0^{\text{co},\rho}$ are plainly broadcasted via BC and because rebroadcasts and majority votes ensure consistency regardless of the behavior of protocol π^{it}.

3.2 Computing the output: The CO secure Protocol $\pi^{\text{co},\rho}$

To obtain the final output y of the protocol Π^ρ we execute a CO secure MPC protocol $\pi^{\text{co},\rho}$ among the n parties P_i $(i \in [n])$ and the emulated party P_0. A setup is neces-

[5] The v in π_i^v stands for majority *vote*, as this is essentially the task performed by these protocol machines.

sary for UC-secure MPC without honest majority, so we use the CRS-model where all parties are given a common, public reference string that is ideally chosen from a given distribution, thus circumventing the impossibility results for the plain model [4, 5]. As noted before we require a number of security properties, namely the protocol $\pi^{\text{co},\rho}$ must be CO secure with abort in the presence of $t < n+1-\rho$ actively corrupted parties[6], IT protect the designated party P_0 and ensure fairness for $t < \frac{n}{2}$ and additionally guarantee robustness for $t \leq \rho$.

We define these security requirements for subprotocol $\pi^{\text{co},\rho}$ formally by stating the $n+1$ party functionalities that $\pi^{\text{co},\rho}$ has to implement:

The ideal model $\mathcal{I}_f^{abort,cb}$ models security with agreement on abort and operates as follows: Let $\text{COMMIT}(\cdot)$ and $\text{OPEN}(\cdot)$ denote the respective procedures for an IT hiding commitment scheme. Functionality $\mathcal{I}_f^{abort,cb}$ takes inputs (x_i^{co}, c_i) from parties P_i $(i \neq 0)$ and $((x_1^{\text{it}}, o_1), \dots, (x_n^{\text{it}}, o_n))$ from party P_0. If a party provides no input, a default input is used, e.g. $(0, 0)$. $\mathcal{I}_f^{abort,cb}$ now checks if $\text{OPEN}(c_i, o_i) = x_i^{\text{it}}$. For every i where this check fails, $\mathcal{I}_f^{abort,cb}$ makes a callback to party P_i asking it to send a new x_i^{it} (use $x_i^{\text{it}} = 0$ in case the party does not reply). The functionality then computes $x_i := x_i^{\text{it}} \oplus x_i^{\text{co}}$ and $y := f(x_1, ..., x_n)$. The result y is output to the adversary, which may now decide to have $\mathcal{I}_f^{abort,cb}$ deliver the output y to the honest parties (abort flag 0) or abort the protocol and have $\perp$ delivered to the honest parties (abort flag 1).

The ideal functionality $\mathcal{I}_f^{des,fair,cb}$ models fairness with a designated party P_0. As such $\mathcal{I}_f^{des,fair,cb}$ only specifies security requirements if P_0 is honest. If P_0 is corrupted control is turned over to the adversary. In case P_0 is honest the functionality $\mathcal{I}_f^{des,fair,cb}$ behaves identically to $\mathcal{I}_f^{abort,cb}$ with two notable exceptions: Inputs and answers to callbacks from all parties P_i $(i \neq 0)$ are directly handed to the simulator (only the privacy of the designated party P_0 is protected), and the simulator must give the abort flag before it receives the output y obtaining y only if it gives flag 0 and thus the honest parties obtain y as well (fairness).

The ideal functionality $\mathcal{I}_f^{des,rob,cb}$ models robustness with a designated party P_0. It only differs from $\mathcal{I}_f^{des,fair,cb}$ in that no abort flag is taken, rather the output y is directly delivered to all parties.

Lemma 3 *For any $\rho < \frac{n}{2}$ there is a protocol $\pi^{\text{co},\rho}$ that implements*

1. *the ideal model $\mathcal{I}_f^{abort,cb}$ (with agreement on abort) with CO security for*[6] $t < n+1-\rho$,
2. *the ideal model $\mathcal{I}_f^{des,fair,cb}$ (with designated party P_0 and fairness) with IT security for*[7] $t < \frac{n}{2}$,
3. *the ideal model $\mathcal{I}_f^{des,rob,cb}$ (with designated party P_0 and robustness) with IT security for $t \leq \rho$*

[6] In the context of the CO protocol $\pi^{\text{co},\rho}$, we consider a set of corrupted parties $\mathfrak{A} \subseteq [n] \cup \{0\}$ that may hence encompass up to $n+1$ parties.

[7] We actually achieve this for $t < n+1-\rho$, but the result for $t < \frac{n}{2}$ suffices subsequently.

from a complete and synchronous network of secure channels, and an authenticated BC channel, and a CRS-setup, in the UC setting, where t is the number of (actively) corrupted parties.

We show that the MPC protocol of [5] can be modified to satisfy Lemma 3 above. Like [15] the MPC protocol in [5] utilizes oblivious transfer (OT) for multiplications and uses the compiler of [15] 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 and then argue why the resulting protocol satisfies Lemma 3. Due to space limitations we refrain from providing a formal proof (for more details, see full version).

Our modifications to [5] encompass the sharing used, the ouput reconstruction, and the CO primitives underlying [5]. We replace the XOR-sharing used for inputs and intermediate results in [5] by a $(2n - \rho)$-out-of-$(2n)$ Shamir-sharing where P_0 receives n shares and each remaining party P_i obtains a single share. Thus we enable any set of $n - \rho + 1$ parties including P_0 to reconstruct the input of up to ρ corrupted parties yielding robustness while maintaining IT privacy for the designated party P_0. We modify the output reconstruction such that designated party P_0 opens its shares only after at least $n - \rho$ parties have correctly opened their shares. In case the designated party P_0 is honest this achieves fairness. Finally, we use CO primitives (commitments, (perfectly) zero-knowledge arguments of knowledge (ZK-AoK), CO zero-knowledge proofs of knowledge (cZK-PoK), OT), which IT protect the designated party P_0. Before we provide a proofsketch for Lemma 3, we discuss our modifications to the computational primitives underlying [5] in more detail:

Oblivious Transfer. As shown in [5, Section 4.1.1] the OT protocol of [15, 14, pp. 640–643] is UC-secure. Furthermore it is easy to see that it IT protects the receiver. The [5] protocol makes no restriction as to who is the sender and who the receiver in any application of OT. 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 [20, Theorem 4.1]. Thus we can IT protect P_0 in any OT invocation.

Commitment. We use the UC-secure IT hiding and IT binding commitment schemes in the CRS-model described in [10] and 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. One-to-many commitment is obtained from the two party commitment of [10] by simply broadcasting commit message and opening as described in [5]. Thus we obtain a UC-secure realization of the one-to-many commiment functionality $\mathcal{I}_{Com,1:M}$ that additionally IT protects the designated party P_0.

ZK proofs. [5] shows how to UC-securely implement the ZK functionality $\mathcal{I}_{ZK,1:M}$ from the commitment functionality $\mathcal{I}_{Com,1:M}$ without use of further CO assumptions. The ZK protocol of [5] is based on the Hamiltonian Cycles ZK proof. 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. Thus by instantiating the one-to-many commitment functionality $\mathcal{I}_{Com,1:M}$ as indicated above, we may obtain a UC-secure realization of $\mathcal{I}_{Com,1:M}$ which additionally IT protects the designated party P_0.

Commit-and-Prove. Instead of directly working with commitment and ZK functionalities, [5] introduces a new primitive called one-to-many commit-and-prove $\mathcal{I}_{CP,1:M}$. [5] exhibits an implementation of $\mathcal{I}_{CP,1:M}$ UC-secure against static adversaries that relies only on $\mathcal{I}_{ZK,1:M}$ and a standard commitment scheme. Implementing $\mathcal{I}_{ZK,1:M}$ as described above we can then use IT hiding or IT binding commitments in the implementation of $\mathcal{I}_{CP,1:M}$ to IT protect the designated party P_0. Further CO assumptions are not required and we can thus implement the commit-and-prove functionality $\mathcal{I}_{CP,1:M}$ UC-securely while IT protecting the designated party P_0.

Proofsketch for Lemma 3 We will now show that the modified version of [5] described above fulfills the requirements stated in Lemma 3.

CO UC security with abort for $t < n + 1 - \rho$ corrupted parties. This first point of Lemma 3 is already implied by [5]. Our modifications to CO primitives and opening procedure 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 [14]. As shares observed by corrupted parties are still uniformly random for $t < n + 1 - \rho$ corrupted parties the modifications to the simulator remain trivial. As such the proof in [5] remains applicable with minimal modifications and we obtain a CO UC-secure implementation of $\mathcal{I}_f^{abort,cb}$.

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.

IT UC security with fairness and designated party P_0 for $t < \frac{n}{2}$ corrupted parties. We sketch a simulator to demonstrate that [5], tweaked as described above, UC-securely implements the ideal model $\mathcal{I}_f^{des,fair,cb}$ (with fairness, designated party P_0 and for at most $t < \frac{n}{2}$ corrupted parties) with IT security as described in point 2 of Lemma 3.

By definition of $\mathcal{I}_f^{des,fair,cb}$, we only have to consider the case where the designated party P_0 is honest, as otherwise no demands are made. The simulator will receive the inputs of all honest parties except P_0 from the ideal functionality $\mathcal{I}_f^{des,fair,cb}$. Furthermore corrupted parties have to commit to their input using binding UC commitments[8], so by extractability the simulator can extract their inputs and forward them to $\mathcal{I}_f^{des,fair,cb}$. 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 $\mathcal{I}_f^{des,fair,cb}$. If a callback to a corrupted party is received from $\mathcal{I}_f^{des,fair,cb}$, the simulator produces the corresponding protocol callback in

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

the simulation (using the equivocability of commitments if needed). Possible answers to callbacks are handled in the same way as inputs. The simulation then proceeds up to the point where the result is opened. The designated party P_0 which is honest by assumption and thus simulated internally by the simulator, is supposed to broadcast its opening information last. If at least $t - \rho$ corrupted parties broadcast their opening information correctly, then the simulator makes use of the equivocability of commitments to have the internally simulated P_0 open the final result to y and sets the abort flag to 0, otherwise it sets the abort flag to 1.

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 the callback and opening phases respectively.

IT UC security with robustness and designated party P_0 for $t \leq \rho$ corrupted parties. We use the exact same simulator as above and note that due to $t - \rho \leq 0$ the simulator will always set the abort flag to 0.

4 Proof of Security of Π^ρ

We have seen how the subprotocols π^{it} and $\pi^{\text{co},\rho}$ can be realized in the UC setting with the required security properties, and how the complete protocol Π^ρ is constructed based on these subprotocols.

We now prove Theorem 1, namely that given subprotocols $\pi^{\text{co},\rho}$ and π^{it}, secure according to Lemma 3 and Lemma 2 respectively, the protocol Π^ρ securely computes the function f, with robust IT security for $t \leq \rho$, with fair IT security for $t < \frac{n}{2}$, and with CO security and agreement on abort for $t < n-\rho$. In the following we denote by R the underlying resources, i.e. a complete and synchronous network of secure channels, an authenticated BC channel, and a CRS-setup. For simplicity, we will generally omit the CRS-setup from any drawings.

4.1 Proof of CO Security with Agreement on Abort for $t < n - \rho$

We begin by proving the first point of Theorem 1, claiming that the protocol Π^ρ implements the ideal model $\mathcal{I}_f^{abort}$ (with agreement on abort) with CO security, for $t < n - \rho$ corrupted parties. The proof is illustrated in Figure 1.

Making use of the UC composition theorems we first replace the IT protocol π^{it} with an ideal system $\mathcal{I}_0^{NoSec}$ which completely turns control over to the adversary (respectively the corresponding simulator σ^N). This makes sense due to the fact that we may have $t \geq \frac{n}{2}$ and thus protocol π^{it} may be completely insecure. As $\mathcal{I}_0^{NoSec}$ only forwards messages to the adversary, we may replace it with corresponding channels.

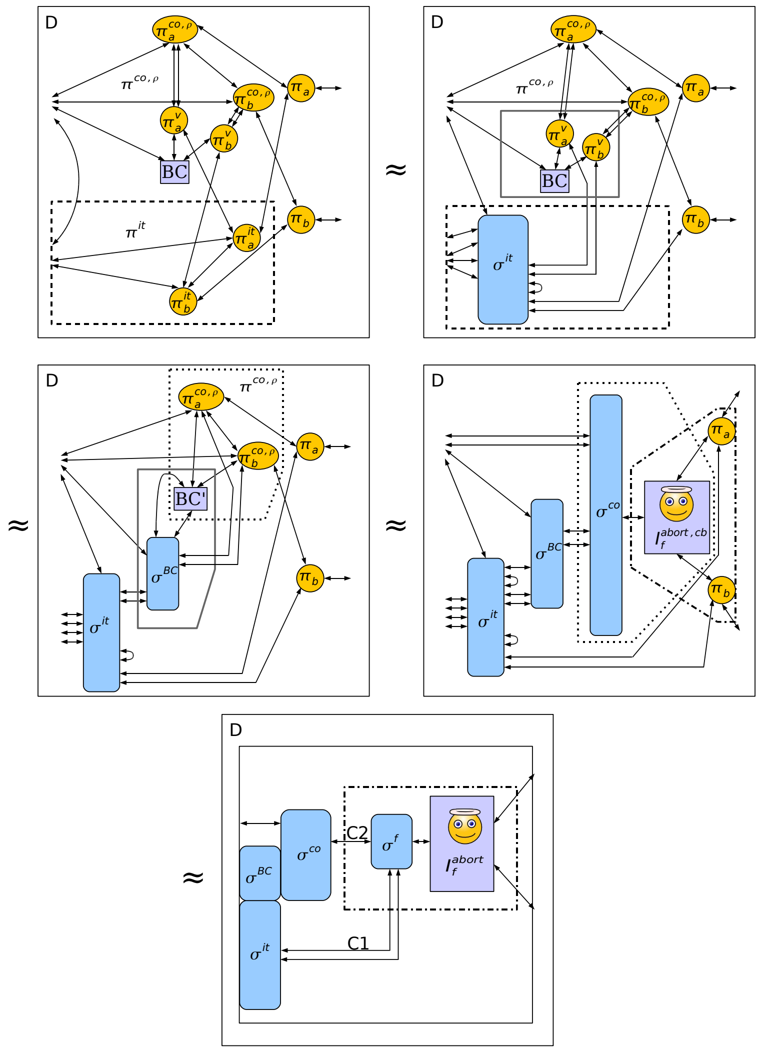

Fig. 1. Proof of security of Π^ρ, CO case.

In a second step, we replace the n party BC channel BC and the protocol machines π_i^v with an $n+1$ party BC channel BC'. The corresponding simulator σ^{BC} internally simulates copies $\widetilde{BC}$ of BC and $\widetilde{\pi_i^v}$ of π_i^v ($i \in \mathfrak{H}$). The $\widetilde{\pi_i^v}$ are connected to $\widetilde{BC}$ as in the protocol. The remaining interfaces of the $\widetilde{\pi_i^v}$ (in particular all private channel interfaces) and of $\widetilde{BC}$ are open to the distinguisher (σ^{BC} forwards all messages), with exception of the interfaces of the $\widetilde{\pi_i^v}$ usually connected to the BC interfaces of the $\pi_i^{\text{co},\rho}$. Now when a BC message is received from an honest $\pi_i^{\text{co},\rho}$ via BC' the simulator σ^{BC} inputs it into $\widetilde{\pi_i^v}$ (which in turn forwards it to $\widetilde{BC}$). On the other hand, when the $\widetilde{\pi_i^v}$ output m as BC message from a corrupted party P_j, this means P_j has sent a BC message and simulator σ^{BC} forwards m to BC' via P_js interface. It is fairly straightforward to see that real system and simulation are perfectly indistinguishable.

The replacements in the previous sections result in a system that applies the protocol $\pi^{\text{co},\rho}$ according to its definition, i.e. there is an $n+1$ party BC channel BC', a CRS-setup, and correct protocol machines for all honest parties (*not* including the emulated party P_0). From the security of $\pi^{\text{co},\rho}$ as stated in Lemma 3 we have a simulator σ^C for $\pi^{\text{co},\rho}$ and we may securely replace the protocol $\pi^{\text{co},\rho}$ with the functionality $\mathcal{I}_f^{abort,cb}$.

As a last step we have to show that the protocol machines π_i implement $\mathcal{I}_f^{abort}$ from functionality $\mathcal{I}_f^{abort,cb}$. Once again we do so by providing an appropriate simulator σ^f.

The simulator σ^f is connected to the interfaces of the corrupted parties to the ideal functionality $\mathcal{I}_f^{abort}$. Furthermore, σ^f has to simulate the communication of π_i $(i \in \mathfrak{H})$ with the party P_0 (or rather the protocol machine π_i^{it} emulating it), which we regard as corrupted in this context. So, the simulator σ^f has two sets of interfaces to the environment, one (C2) corresponding to the corrupted protocol machines $\pi_i^{\text{co},\rho}$ of the protocol $\pi^{\text{co},\rho}$ and one (C1) corresponding to the interfaces of the protocol machines π_i of the honest parties to the designated party P_0 (Figure 1). The simulator has to produce the correct distributions on these two interfaces and the interface to the ideal functionality $\mathcal{I}_f^{abort}$, in order to render $\sigma^f(\mathcal{I}_f^{abort})$ indistinguishable from $\pi_{\mathfrak{H}}(\mathcal{I}_f^{abort,cb})$.

First, over interface C1, the distinguisher D expects the input of the honest parties intended for the emulated party P_0, which is of the form $\{x_i^{\text{it}}, o_i\}_{i\in\mathfrak{H}}$. This information is not available and has to be simulated by σ^f correctly. Over interface C2, the σ^f receives the input of the corrupted parties intended for $\mathcal{I}_f^{abort,cb}$, which consists of two parts: $\{x_i^{\text{it}}, o_i\}_{i\in[n]}$ representing the input of party P_0 to protocol $\pi^{\text{co},\rho}$, and $\{x_i^{\text{co}}, c_i\}_{i\in\mathfrak{A}}$ representing the input of the corrupted parties $P_i, i \in \mathfrak{A}$ to protocol $\pi^{\text{co},\rho}$.

The simulator σ^f performs the following steps:

1. Choose $\tilde{x}_i^{\text{it}}$ $(i \in \mathfrak{H})$ for the honest parties at random and compute IT hiding commitments $[\tilde{c}_i, \tilde{o}_i] = \text{COMMIT}(\tilde{x}_i^{\text{it}})$.
2. Give $\{\tilde{x}_i^{\text{it}}, \tilde{o}_i\}_{i\in\mathfrak{H}}$ as output over C1, intended for P_0.
3. Receive two kinds of input over C2, intended for $\mathcal{I}_f^{abort,cb}$:
   – $\{x_i^{\text{it}}, o_i\}_{i\in[n]}$ representing the input of party P_0 to protocol $\pi^{\text{co},\rho}$.
   – $\{x_i^{\text{co}}, c_i\}_{i\in\mathfrak{A}}$ representing the input of the corrupted $P_i, i \in \mathfrak{A}$ to protocol $\pi^{\text{co},\rho}$.
4. Perform the CHECK: $\forall i \in \mathfrak{A} : \text{OPEN}(c_i, o_i) \equiv x_i^{\text{it}}$ (correct opening information)
   – [holds] For every triple, where the check holds, use the existing values for x_i^{it}.
   – [fails] For every triple, where the check fails, perform a callback and use the new values for x_i^{it}. Use $x_i^{\text{it}} = 0$ in case the callback is not answered.
5. Compute $x_i = x_i^{\text{co}} \oplus x_i^{\text{it}}$, $i \in \mathfrak{A}$ and input the x_i to $\mathcal{I}_f^{abort}$
6. Forward the output y of $\mathcal{I}_f^{abort}$ to the distinguisher via C2.
7. Forward the abort flag from C2 to $\mathcal{I}_f^{abort}$

We now argue that the simulator σ^f indeed renders $\sigma^f(\mathcal{I}_f^{abort})$ indistinguishable from $\pi_{\mathfrak{H}}(\mathcal{I}_f^{abort,cb})$.

Protocol machine π_i in $\pi_{\mathfrak{H}}(\mathcal{I}_f^{abort,cb})$ first splits its input into $x_i = x_i^{\text{co}} \oplus x_i^{\text{it}}$ (where x_i^{it} is uniformly random) and computes the IT hiding commitment $[c_i, o_i] = \text{COMMIT}(x_i^{\text{it}})$. Then, π_i provides (x_i^{it}, o_i) as input to protocol π^{it}, which may be insecure and controlled by the adversary. As such we think of the input of the honest parties to π^{it} as being directly forwarded to the adversary (in other words, the environment expects these values over C1). σ^f simulates this indistinguishably by providing random values $\{(\tilde{x}_i^{\text{it}}, \tilde{o}_i)\}_{i\in\mathfrak{H}}$ with appropriate opening information to the environment over C1.

After that, protocol machine π_i provides x_i^{co} together with the commitment c_i as input to the ideal functionality $\mathcal{I}_f^{abort,cb}$, while the the protocol π^{it} is to provide the input $\{x_i^{\text{it}}, o_i\}_{i\in[n]}$. $\mathcal{I}_f^{abort,cb}$ checks if $\text{OPEN}(c_i, o_i) \equiv x_i^{\text{it}}$ and issues a callback for any i where the check fails. σ^f simulates this behavior identically for all corrupted parties. The ideal functionality $\mathcal{I}_f^{abort,cb}$ then computes f on the $x_i = x_i^{\text{co}} \oplus x_i^{\text{it}}$, which simulator σ^f simulates by inputting the x_i to $\mathcal{I}_f^{abort}$. 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^{it} (which being CO bounded it cannot).[9]

Finally $\mathcal{I}_f^{abort,cb}$ delivers the output y to the adversary and awaits an abort flag deciding output delivery to honest parties. Outputs are simply forwarded by π_i. Functionality $\mathcal{I}_f^{abort}$ behaves identically, and as such the simulator σ^f need only forward the messages in question.

Hence the protocol Π^ρ constitutes a CO secure implementation of the functionality $\mathcal{I}_f^{abort}$ as claimed in the first point of Theorem 1.

4.2 Proof of IT Security with Fairness for $t < \frac{n}{2}$

We continue by proving the second point of Theorem 1, claiming that the protocol Π^ρ implements the ideal model $\mathcal{I}_f^{fair}$ (with fairness) with IT security, for $t < \frac{n}{2}$ corrupted parties. The proof is illustrated in Figure 2.

Making use of the UC composition theorems we first replace the IT protocol π^{it} with the ideal system $\mathcal{I}_0^{sec}$ which it implements IT securely with simulator σ^N by definition and according to Lemma 2.

We now have an ideal functionality $\mathcal{I}_0^{sec}$, an n party BC channel BC, and protocol machines π_i^v. As it is our eventual goal to replace the subprotocol $\pi^{\text{co},\rho}$, we first have to replace these components with an $n+1$ party BC channel BC' and the protocol machine $\pi_0^{\text{co},\rho}$.

The corresponding simulator σ^{BC} internally simulates copies $\widetilde{BC}$ of BC and $\widetilde{\pi_i^v}$ of π_i^v $(i \in \mathfrak{H})$. The $\widetilde{\pi_i^v}$ are connected to $\widetilde{BC}$ as in the protocol. The interfaces of $\widetilde{BC}$ to corrupted parties are open to the distinguisher (σ^{BC} forwards all messages). Now when a BC message m is received from $\pi_0^{\text{co},\rho}$ via BC' the simulator σ^{BC} simulates the actions of $\mathcal{I}_0^{sec}$ by outputting m to the corrupted parties and inputting it to the $\widetilde{\pi_i^v}$. When

[9] The commitments to the x_i^{it} and the callbacks guarantee that the computation in protocol $\pi^{\text{co},\rho}$ is carried out with correct values x_i^{it}. That is, the input x_i^{co} provided by the (honest) parties and the inputs x_i^{it} provided by party P_0 have the relation $x_i^{\text{co}} \oplus x_i^{\text{it}} = x_i$. Otherwise, if the adversary controls the protocol π^{it} (as is the case here), he could manipulate the values x_i^{it} leading to a computation with wrong inputs x_i and hence to an incorrect result.

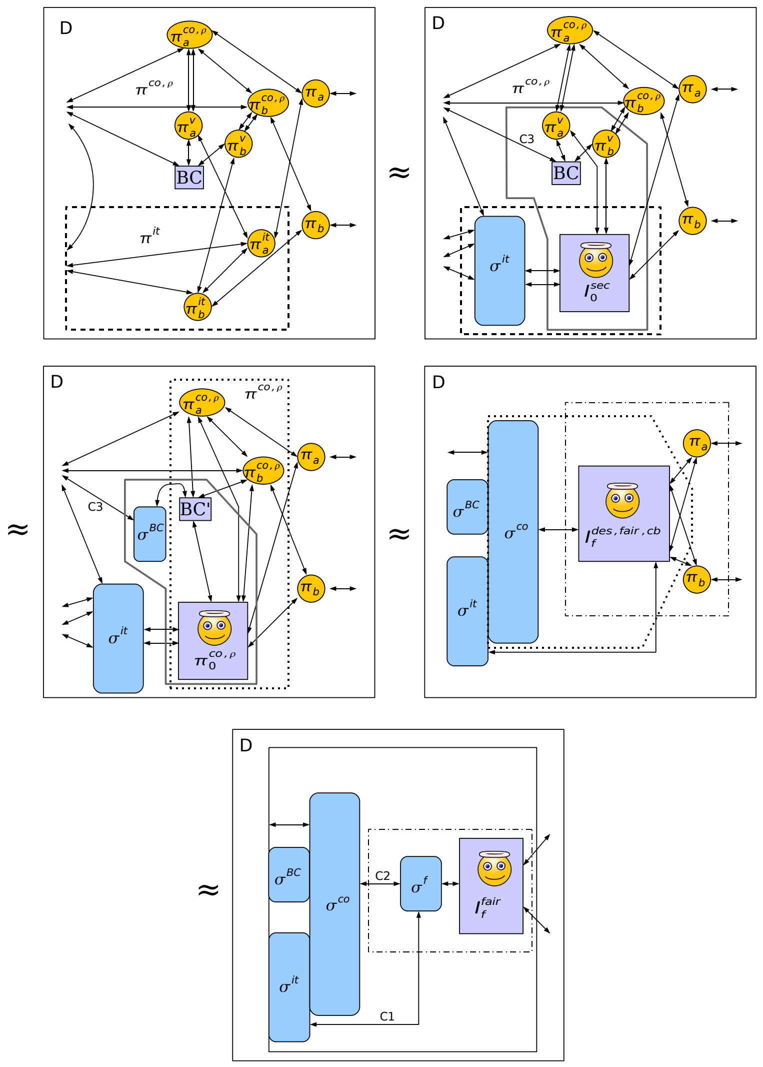

Fig. 2. Proof of security of Π^ρ, IT case.

a BC message is received from an honest $\pi_i^{\text{co},\rho}$ via BC' the simulator σ^{BC} inputs it into $\widetilde{\pi_i^v}$ (which in turn forwards it to $\widetilde{BC}$). On the other hand, when the $\widetilde{\pi_i^v}$ output m as BC message from a corrupted party P_j, this means P_j has sent a BC message and σ^{BC} forwards m to P_js interface to BC'. It is fairly straightforward to see that real system and simulation are perfectly indistinguishable.

After this replacement, we have an instance of protocol $\pi^{\text{co},\rho}$, including a correct protocol machine $\pi_0^{\text{co},\rho}$ for the designated party P_0 and utilizing an $n + 1$ party BC channel BC' and a CRS-setup. We now replace the protocol $\pi^{\text{co},\rho}$ with the ideal model $\mathcal{I}_f^{des,fair,cb}$ which it IT securely implements as stated in point 2 of Lemma 3. The corresponding simulator we call σ^C.

It now remains to show that the protocol machines π_i IT securely implement $\mathcal{I}_f^{fair}$ from $\mathcal{I}_f^{des,fair,cb}$. Once again we do so by providing an appropriate simulator σ^f that renders $\sigma^f(\mathcal{I}_f^{fair})$ indistinguishable from $\pi_{\mathfrak{H}}(\mathcal{I}_f^{des,fair,cb})$.

As each π_i provides two inputs (x_i^{it}, o_i) and (x_i^{co}, c_i) to $\mathcal{I}_f^{des,fair,cb}$, we can think of the simulator σ^f as simulating two sets of interfaces of $\mathcal{I}_f^{des,fair,cb}$ to the environment. By C1 we denote the t interfaces associated with the designated party P_0 which carry the inputs $\{x_i^{\text{it}}, o_i\}_{i\in\mathfrak{A}}$ of the corrupted parties. By C2 we denote the t interfaces corresponding to the $\{x_i^{\text{co}}, c_i\}_{i\in\mathfrak{A}}$ inputs of corrupted parties, as well as leakage and possible callbacks and replies.

The simulator σ^f operates as follows:

1. Choose $\tilde{x}_i^{\text{co}}$ and $\tilde{x}_i^{\text{it}}$ $(i \in \mathfrak{H})$ for the honest parties at random and compute IT hiding commitments $[\tilde{c}_i, \tilde{o}_i] = \text{COMMIT}(\tilde{x}_i^{\text{it}})$.
2. Output $\{\tilde{x}_i^{\text{co}}, \tilde{c}_i\}_{i\in\mathfrak{H}}$ over C2 (information leakage).
3. Receive inputs $\{x_i^{\text{it}}, o_i\}_{i\in\mathfrak{A}}$ over C1 and $\{x_i^{\text{co}}, c_i\}_{i\in\mathfrak{A}}$ over C2
4. Perform the CHECK: $\forall i \in \mathfrak{A} : \text{OPEN}(c_i, o_i) \equiv x_i^{\text{it}}$ (correct opening information)
   - [holds] For every triple, where the check holds, use the existing values for x_i^{it}.
   - [fails] For every triple, where the check fails, perform a callback and use the new values for x_i^{it}. Use $x_i^{\text{it}} = 0$ in case the callback is not answered.
5. Compute $x_i = x_i^{\text{co}} \oplus x_i^{\text{it}}$, $i \in \mathfrak{A}$ and input the x_i to $\mathcal{I}_f^{fair}$
6. Forward the abort flag from C2 to $\mathcal{I}_f^{fair}$
7. Forward the output y of $\mathcal{I}_f^{fair}$ to C2.

It only remains show that the simulator σ^f renders the ideal setting $\sigma^f(\mathcal{I}_f^{fair})$ indistinguishable from $\pi_{\mathfrak{H}}(\mathcal{I}_f^{des,fair,cb})$:

In addition to (x_i^{co}, c_i), protocol machine π_i inputs (x_i^{it}, o_i). $\mathcal{I}_f^{des,fair,cb}$ checks if $\text{OPEN}(c_i, o_i) \equiv x_i^{\text{it}}$ and issues a callback for any i where the check fails. Simulator σ^f receives these values from the corrupted parties and simulates this behavior identically. The ideal functionality $\mathcal{I}_f^{des,fair,cb}$ then computes f on the $x_i = x_i^{\text{co}} \oplus x_i^{\text{it}}$, which σ^f simulates by inputting the x_i to $\mathcal{I}_f^{fair}$.

Finally $\mathcal{I}_f^{des,fair,cb}$ awaits an abort flag and delivers output if the flag is 0. Outputs are simply forwarded by π_i. $\mathcal{I}_f^{abort}$ behaves identically, and as such the simulator σ^f need only forward the messages in question.

Hence the protocol Π^ρ constitutes an IT secure implementation of the functionality $\mathcal{I}_f^{fair}$ as claimed in the second point in Theorem 1.

4.3 Proof of IT Security with Robustness for $t \le \rho$

The proof in this case is essentially indentical to the proof for the case with fairness above. One simply replaces $\mathcal{I}_f^{fair}$ with $\mathcal{I}_f^{sec}$ and $\mathcal{I}_f^{des,fair,cb}$ with $\mathcal{I}_f^{des,rob,cb}$, removes the treatment of the abort flag from σ_f and relies on point 3 instead of point 2 of Lemma 3 for introducing $\mathcal{I}_f^{des,rob,cb}$.

5 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 $ExtConsBC$ of [11]. For two thresholds t_v and t_c where $t_v \leq t_c$, and either $t_v = 0$ or $t_v + 2t_c < n$ $ExtConsBC$ delivers a robust BC for $t \leq t_v$ and a BC with fairness (but without robustness) for $t_v < t \leq t_c$. More to the point $ExtConsBC$ 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 robustness bound $\rho > 0$, setting $t_v = \rho < \frac{n}{3}$ and $t_c = \lceil \frac{n-t_v}{2} \rceil - 1$ we achieve full IT security (with robustness) for $t \leq \rho$ and IT security with fairness (no robustness) for $t < \frac{n-\rho}{2}$. Unfortunately these results do not (and connot) go beyond those of [11] which they have proven optimal for this case.

For robustness bound $\rho = 0$ however, setting $t_v = \rho = 0$ and $t_c = n$ we achieve IT security with fairness (no robustness) for $t < \frac{n}{2}$ and CO security with agreement on abort for $t < n$. This result is new and actually matches the result for $\rho = 0$ according to Theorem 1 in the case where a BC channel is provided.

6 Conclusions

We described a protocol Π^ρ that provides a flexible and optimal trade-off between full IT security (with robustness), IT fairness (no robustness), and CO security with abort (no fairness). More precisely, for an abitrarily chosen robustness parameter $\rho < \frac{n}{2}$, the protocol Π^ρ is fully IT secure for $t \leq \rho$, IT fair for $t < \frac{n}{2}$, and CO secure with agreement on abort for $t < n - \rho$ corrupted parties. These results are optimal with respect to the bounds stated in [9, 17, 16].

We proved the UC security of Π^ρ in the synchronous secure channels model with BC and a CRS. In the full version of this paper we also show a simple variation of Π^ρ that relies on [15] instead of [5] 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 [11] 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.

References

1. M. Backes, B. Pfitzmann, and M. Waidner. A general composition theorem for secure reactive systems. In *TCC'04*, volume 2951 of *LNCS*, pages 336–354. Springer, 2004.
2. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC'88*, pages 1–10. ACM, 1988.
3. R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. *FOCS'01*, pages 136–145, 2001.
4. R. Canetti and M. Fischlin. Universally composable commitments. In *CRYPTO'01*, pages 19–40. Springer, 2001.

5. R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *STOC'02*, pages 494–503. ACM, 2002.
6. D. Chaum. The spymasters double-agent problem: Multiparty computations secure unconditionally from minorities and cryptographically from majorities. In *CRYPTO'89*, pages 591–602. Springer, 1989.
7. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *STOC'88*, pages 11–19. ACM, 1988.
8. D. Chaum, I. Damgård, and J. Graaf. Multiparty computations ensuring privacy of each party's input and correctness of the result. In *CRYPTO'87*, pages 87–119. Springer, 1988.
9. R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *STOC'86*, pages 364–369. ACM, 1986.
10. I. Damgård and J. B. Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *CRYPTO'02*, volume 2442 of *LNCS*, pages 581–596. Springer, 2002.
11. M. Fitzi, M. Hirt, T. Holenstein, and J. Wullschleger. Two-threshold broadcast and detectable multi-party computation. In *EUROCRYPT'03*, volume 265 of *LNCS*, pages 51–67. Springer, May 2003.
12. M. Fitzi, T. Holenstein, and J. Wullschleger. Multi-party computation with hybrid security. In *EUROCRYPT'04*, volume 3027 of *LNCS*, pages 419–438. Springer, May 2004.
13. O. Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, 2001.
14. O. Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, 2004.
15. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC'87*, pages 218–229. ACM, 1987.
16. Y. Ishai, E. Kushilevitz, Y. Lindell, and E. Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In *CRYPTO'06*, volume 4117/2006, pages 483–500. Springer, 2006.
17. J. Katz. On achieving the "best of both worlds" in secure multiparty computation. In *STOC'07*, pages 11–20. ACM, 2007.
18. J. Kilian. More general completeness theorems for secure two-party computation. In *STOC'00*, pages 316–324. ACM, 2000.
19. T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *STOC'89*, pages 73–85. ACM, 1989.
20. J. Wullschleger. *Oblivious-Transfer Amplification*. PhD thesis, ETH Zürich, 2007.
21. A. C. Yao. Protocols for secure computations (extended abstract). In *FOCS'82*, pages 160–164. IEEE, 1982.