Passive Corruption in Statistical Multi-Party Computation*

Martin Hirt[1], Christoph Lucas[1], Ueli Maurer[1], and Dominik Raub[2]

[1] Department of Computer Science, ETH Zurich, Switzerland
{hirt, clucas, maurer}@inf.ethz.ch
[2] Department of Computer Science, University of Århus, Denmark
raub@cs.au.dk

Abstract. The goal of *Multi-Party Computation* (MPC) is to perform an arbitrary computation in a distributed, private, and fault-tolerant way. For this purpose, a fixed set of n parties runs a protocol that tolerates an adversary corrupting a subset of the parties, preserving certain security guarantees like correctness, secrecy, robustness, and fairness. Corruptions can be either *passive* or *active*: A passively corrupted party follows the protocol correctly, but the adversary learns the entire internal state of this party. An actively corrupted party is completely controlled by the adversary, and may deviate arbitrarily from the protocol. A *mixed adversary* may at the same time corrupt some parties actively and some additional parties passively.
In this work, we consider the statistical setting with mixed adversaries and study the exact consequences of active and passive corruptions on secrecy, correctness, robustness, and fairness separately (i.e., hybrid security). Clearly, the number of passive corruptions affects the thresholds for secrecy, while the number of active corruptions affects all thresholds. It turns out that in the statistical setting, the number of passive corruptions in particular also affects the threshold for correctness, i.e., in all protocols there are (tolerated) adversaries for which a single additional passive corruption is sufficient to break correctness. This is in contrast to both the perfect and the computational setting, where such an influence cannot be observed. Apparently, this effect arises from the use of information-theoretic signatures, which are part of most (if not all) statistical protocols.

Keywords: Multi-party computation, passive corruption, statistical security, hybrid security, mixed adversaries.

1 Introduction

1.1 Secure Multi-Party Computation

Multi-Party Computation (MPC) allows a set of n parties to securely perform an arbitrary computation in a distributed manner, where security means that secrecy of the inputs and correctness of the output are maintained even when some of the parties are dishonest. The dishonesty of parties is modeled with a central adversary who corrupts parties. The adversary can be *passive*, i.e. can read the internal state of the corrupted parties, or *active*, i.e., can make the corrupted parties deviate arbitrarily from the protocol.

MPC was originally proposed by Yao [Yao82]. The first general solution was provided in [GMW87], where, based on computational intractability assumptions, security against a passive adversary was achieved for $t < n$ corruptions, and security against an active adversary was achieved for $t < \frac{n}{2}$ corruptions. Information-theoretic security was achieved in [BGW88, CCD88] at the price of lower corruption thresholds, namely $t < \frac{n}{2}$ for passive and $t < \frac{n}{3}$ for active adversaries. The latter bound can be improved to $t < \frac{n}{2}$ if both broadcast channels are assumed and a small error probability is tolerated [RB89, Bea89]. These results were generalized to the non-threshold setting, where the corruption capability of the adversary is not specified by a threshold t, but rather by a so called adversary structure $\mathcal{Z}$, a monotone collection of subsets of the player set, where the adversary can corrupt the players in one of these subsets [HM97].

All mentioned protocols achieve full security, i.e. secrecy, correctness, and robustness. *Secrecy* means that the adversary learns nothing about the honest parties' inputs and outputs

* An extended abstract of this paper appeared at ICITS 12 [HLMR12]. This work was partially supported by the Zurich Information Security Center.

(except, of course, for what can be derived from the corrupted parties' inputs and outputs). *Correctness* means that all parties either output the right value or no value at all. *Robustness* means that the adversary cannot prevent the honest parties from learning their respective outputs. This last requirement turns out to be very demanding. Therefore, relaxations of full security have been proposed, where robustness is replaced by weaker output guarantees: *Fairness* means that the adversary can possibly prevent the honest parties from learning their outputs, but then also the corrupted parties do not learn their outputs. *Agreement on abort* means that the adversary can possibly prevent honest parties from learning their output, even while corrupted parties learn their outputs, but then the honest parties at least reach agreement on this fact (and typically make no output). In our constructions, all abort decisions are based on publicly known values. Hence, we have agreement on abort for free.[3]

The traditional setting of MPC has been generalized in two directions. On the one hand, the notion of *hybrid security* was introduced to allow for protocols with different security guarantees depending on the number of corrupted parties [Cha89, FHHW03, FHW04, IKLP06, Kat07, LRM10, HLMR11]. Intuitively, the more corrupted parties, the less security is guaranteed. This model also allows to analyze each security guarantee separately and independent of other guarantees. On the other hand, protocols were presented that do not restrict the adversary to a single corruption type [Cha89, DDWY93, FHM98, FHM99, BFH+08, HMZ08, HLMR11]. The *mixed adversaries* considered there can perform each corruption with one out of several corruption types. This allows to consider e.g. active and passive corruption in the same protocol execution.

1.2 Contributions

In this work, we consider a setting with mixed adversaries and hybrid security. This allows, for the first time, to separately analyze the relation between passive corruption and the various security guarantees. It turns out that, in the statistical model, passive corruption does not only affect secrecy, but in particular also correctness. In most statistically secure protocols, some kind of information-theoretic signature is used. When combining active and passive corruptions, one inherent problem of any kind of information-theoretic signature is that passively corrupted parties cannot reliably verify signed values. Existing protocols for the statistical setting assume an honest majority. Therefore, a simple majority vote on the signature guarantees reliable verification even for passively corrupted parties. In this work, we show that this assumption is too strong, and that signatures can be used even without an honest majority. As the main technical contribution, we provide optimal protocols for both general and threshold adversaries that cope with this issue. As a new technique for the setting with general adversaries, we introduce *group commitments*, a non-trivial extension of IC-Signatures, which might be of independent interest.

Furthermore, we introduce the notion of *multi-thresholds*. To the best of our knowledge, all known protocols for threshold mixed adversaries (e.g. [FHM98]) characterize the tolerable adversaries with a single pair of thresholds (one threshold for the number of actively, and one for the number of passively corrupted parties). This pair represents the single maximal adversary that can be tolerated. We generalize this basic characterization to allow for several incomparable maximal adversaries. It turns out that, in our setting, multi-thresholds allow to construct protocols that tolerate strictly more adversaries than a single pair of thresholds, without losing efficiency.

[3] The impossibility proof holds even when agreement on abort is not required.

1.3 Model

We consider n parties $p_1, \ldots, p_n$, connected by pairwise synchronous secure channels and authenticated broadcast channels[4], who want to compute some probabilistic function over a finite field $\mathbb{F}$, represented as circuit with input, addition, multiplication, random, and output gates. This function can be reactive, where parties can provide further inputs after having received some intermediate outputs.

There is a central adversary with unlimited computing power who corrupts some parties passively (and reads their internal state) or even actively (and makes them misbehave arbitrarily). We denote the actual sets of actively (passively) corrupted parties by $\mathcal{D}^*$ ($\mathcal{E}^*$), where $\mathcal{D}^* \subseteq \mathcal{E}^*$. Uncorrupted parties are called *honest*, non-actively corrupted parties are called *correct*. The security of our protocols is statistical, i.e. information-theoretic with a small error probability. We say a security guarantee holds *statistically* if it holds with overwhelming probability. The guaranteed security properties (secrecy, correctness, fairness, robustness, agreement on abort) depend on $(\mathcal{D}^*, \mathcal{E}^*)$.

For ease of notation, we assume that if a party does not receive an expected message (or receives an invalid message), a default message is used instead. Furthermore, we use subprotocols that might abort. Such an abort is always global, i.e., if any subprotocol aborts, the whole protocol execution halts.

In the analysis of our protocols, we assume "instant randomness", i.e. parties generate their randomness on the fly when needed in the protocol run. This allows even passively corrupted parties to e.g. choose challenges in zero-knowledge proofs that are unpredictable to the adversary. Note that in a setting without secrecy, we have no input independence[5]. Hence, standard techniques (e.g. Blum coin-toss) to jointly generate these challenges are insecure.

1.4 Outline of the paper

The paper is organized as follows: In Sec. 2, we present information checking, which is used as a basic primitive in our protocols. As a main technical contribution, in Sections 3 and 4, we present protocols for the model with mixed adversaries and hybrid security for both general and threshold adversaries, together with optimal bounds. In Sec. 5, we provide conclusions of our results.

2 Information Checking

Information checking (IC) [RB89, CDD+99] is a primitive that allows a sender to send a value to an intermediary, such that when the receiver obtains this value from the intermediary, he can check that this is indeed the value from the sender. When all parties act as receivers, this primitive is called *IC signature*, and the sender is called signer. IC signatures are realized using a pair of protocols IC-SIGN and IC-REVEAL. IC-SIGN allows a signer to sign a value for a particular intermediary (while providing secrecy with respect to the remaining parties), and IC-REVEAL allows this intermediary to verifiably forward this value to all other parties. In the following, we assume that each pair of parties (p_i, p_j) has a value α_{ij} which only they know. This setup can easily be achieved for each pair (p_i, p_j) by having p_i chose and send α_{ij} to p_j before the protocol starts.

Definition 1 (α-consistent). *A triple (v, y, z) is α-consistent, if $z = (y - v)\alpha + v$, i.e. if the points $(0, v)$, $(1, y)$, and (α, z) lie on a line.*

[4] In [PW92] it is shown how broadcast can be implemented given a setup.

[5] That means, the adversary can choose the inputs of actively corrupted parties after learning the inputs of correct parties.

Definition 2 (IC-Signature). *A value v is* IC-signed *(or simply* signed*) by signer p_i for intermediary p_j, denoted by $\langle v \rangle_{i,j}$, if p_j holds values $v, y_1, ..., y_n$ and each $p_k \in \mathcal{P}$ holds a value z_k such that (v, y_k, z_k) is α_{ik}-consistent. In analogy to traditional signatures, we equivalently say that the intermediary p_j* holds *the signature $\langle v \rangle_{i,j}$.*

Note that a default signature $\langle v \rangle_{i,j}$ can be generated given that all parties know the value v. Furthermore, given v, α_{ik}, and z_k, a value y_k can be computed efficiently such that (v, y_k, z_k) are α_{ik}-consistent. This implies in particular that if the intermediary is actively corrupted, then any z-values held by the (correct) recipients constitute a valid signature for v. Additionally, IC-signatures are linear, i.e. the sum of two signatures $\langle v \rangle_{i,j}$ and $\langle v' \rangle_{i,j}$ from signer p_i to intermediary p_j for values v and v', respectively, is a signature from p_i to p_j for the sum $v + v'$.

The IC Sign Protocol. The IC sign protocol assumes that the signer and the intermediary both know a value v, e.g. that the signer has already sent v to the intermediary. The protocol either computes a valid signature on v, or outputs $\perp$ to all parties.

Protocol IC-SIGN: Given a signer p_i and an intermediary p_j that both know a value v, either compute a valid signature $\langle v \rangle_{i,j}$ on this value, or output $\perp$ to all parties.

1. For each recipient $p_k \in \mathcal{P}$:
   (a) p_i chooses v'_k, y_k, y'_k uniformly at random and sets z_k and z'_k such that (v, y_k, z_k) and (v', y'_k, z'_k) are both α_{ik}-consistent. p_i sends v'_k, y_k, y'_k to p_j, and z_k, z'_k to p_k.
   (b) p_k broadcasts a uniform random challenge r_k. Then, both p_i and p_j broadcast $v'' := v'_k + r_k v$ and $y'' := y'_k + r_k y_k$.
   (c) If in the previous step p_i and p_j did not broadcast the same values, all parties output $\perp$.
   (d) p_k broadcasts "accept" if $(v'', y'', z'_k + r_k z_k)$ is α_{ik}-consistent. Otherwise, p_k broadcasts ("reject",α_{ik},z_k), and p_j sets y_k such that (v, y_k, z_k) are α_{ik}-consistent.
2. p_j outputs $v, y_1, \ldots, y_n$, and each p_k outputs z_k.

Fig. 1. The IC-Sign protocol.

Lemma 1. *Given a signer p_i and an intermediary p_j that both know the same value v. If p_i and p_j are correct,* IC-SIGN *correctly computes a valid signature $\langle v \rangle_{i,j}$ on v while providing secrecy with respect to the remaining parties. Otherwise,* IC-SIGN *either correctly computes a valid signature $\langle v \rangle_{i,j}$ on v, or all (correct) parties output $\perp$, with overwhelming probability.* IC-SIGN *is always secret and robust.*

Proof. CORRECTNESS: If p_i and p_j are both correct, it is trivial to see that the output is a valid signature (and not $\perp$). Else, if the intermediary p_j is corrupted, either the output of the correct parties trivially corresponds to a valid signature, or all parties output $\perp$. Otherwise, if the intermediary p_j is correct, we have to show that for all correct receivers p_k it holds that (v, y_k, z_k) is α_{ik}-consistent. If p_k is correct, the adversary does not know r_k in advance, and an inconsistent triple (v, y_k, z_k) would be detected by p_k with overwhelming probability.
SECRECY: For a corrupted p_k, both v'' and y'' look uniformly random. Hence, p_k obtains no information apart from his intended output. Furthermore, the value α_{ik} is broadcasted in Step 1.d) only if p_i or p_k are actively corrupted. Hence, the adversary knew the value already beforehand.
ROBUSTNESS: It follows from inspection that the protocol does not abort. □

The IC Reveal Protocol. If a value v is IC-signed (e.g. if the IC-SIGN protocol resulted in a valid signature and did not terminate with output $\perp$), the IC-REVEAL protocol allows to verifiably reveal the value v to all parties.

Lemma 2. *Given a signature $\langle v \rangle_{i,j}$,* IC-REVEAL *robustly computes the output $x_k \in \{(\text{"accept"}, v'), \text{"reject"}\}$ for each p_k. We have the following correctness guarantees:*

Protocol IC-REVEAL: Given a signature $\langle v \rangle_{i,j}$, reveal a value v' to all parties.

1. p_j broadcasts $(v, y_1, \ldots, y_n)$.
2. Each receiver p_k outputs ("accept", v) if (v, y_k, z_k) is α_{ik}-consistent, and "reject" otherwise.

Fig. 2. The IC-Reveal protocol.

1. *If p_j is correct, all correct parties p_k output $x_k =$ ("accept", v).*
2. *Else, if both p_i and p_k are honest, then $x_k \in \{$("accept", v), "reject"$\}$ (with overwhelming probability, even when p_j is active).*

Note that there is no agreement on the output of correct parties. Furthermore, if p_j is active and p_i or p_k is not honest, then p_k might output $x_k =$ ("accept", v') for $v' \neq v$.

Proof. CORRECTNESS: If p_j is correct, it broadcasts values in Step 1, which are α_{ik}-consistent and hence accepted. If p_j is actively corrupted, but both p_i and p_k are honest, then p_j does not know α_{ik}. Hence, for $v' \neq v$, p_j cannot produce a value y'_k where (v', y'_k, z_k) are α_{ik}-consistent, except with negligible probability.
ROBUSTNESS: It follows from inspection that the protocol does not abort. □

3 MPC with General Adversaries

Traditionally, protocols for general adversaries are characterized by an adversary structure $\mathcal{Z}$ that specifies the tolerated subsets of the player set [HM97]. For our setting, we have to extend this basic representation: On the one hand, we consider mixed adversaries, which are characterized by adversary structures consisting of tuples $(\mathcal{D}, \mathcal{E})$ of subsets of $\mathcal{P}$, where the adversary may corrupt the parties in $\mathcal{E}$ passively, and the parties in $\mathcal{D} \subseteq \mathcal{E}$ even actively. On the other hand, each security guarantee depends on the sets of *actually* corrupted parties $(\mathcal{D}^*, \mathcal{E}^*)$. We consider four security guarantees, namely correctness, secrecy, robustness, and fairness. This is modeled with four adversary structures $\mathcal{Z}^c$, $\mathcal{Z}^s$, $\mathcal{Z}^r$, and $\mathcal{Z}^f$, one for each security requirement[6]: Correctness is guaranteed for $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^c$, secrecy is guaranteed for $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^s$, robustness is guaranteed for $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^r$, and fairness is guaranteed for $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^f$. We have the assumption that $\mathcal{Z}^r \subseteq \mathcal{Z}^c$ and $\mathcal{Z}^f \subseteq \mathcal{Z}^s \subseteq \mathcal{Z}^c$, as secrecy and robustness are not well defined without correctness, and as fairness cannot be achieved without secrecy.

Our protocol for general adversaries is based on [HMZ08], which is an adaptation of the perfectly secure protocol of [Mau02] to the statistical case. For a generic protocol construction, it is sufficient to consider two parameters [HLMR11]: First, the state that is held in the protocol is defined in terms of a parameter that influences the secrecy. This parameter is the sharing parameter $\mathcal{S}$, a collection of subsets of $\mathcal{P}$ that defines which party obtains which values. Second, the reconstruct protocol is expressed in terms of an additional parameter determining the amount of error correction taking place. This parameter is the reconstruction parameter $\mathcal{R}$. In contrast to the perfect case, here we need to consider both active and passive corruption. Therefore, the reconstruction parameter is a monotone collection of pairs $(\mathcal{D}, \mathcal{E})$ of subsets of $\mathcal{P}$ where $\mathcal{D} \subseteq \mathcal{E}$: If all errors can be explained with an adversary $(\mathcal{D}, \mathcal{E}) \in \mathcal{R}$, the errors are corrected and the protocol continues; otherwise it aborts. This implies that the protocol aborts only if the actual adversary is not in $\mathcal{R}$. Such aborts are global, i.e., if some subprotocol aborts, the entire protocol execution halts.

3.1 A Parametrized Protocol for General Adversaries

In the following, we present the parametrized subprotocols for general adversaries and analyze them with respect to correctness, secrecy, and robustness. The main result (including

[6] Since all our protocols achieve agreement on abort for free, we do not introduce a separate structure for this security property.

fairness) is discussed in Sec. 3.2. As a first step, we introduce *group commitments* which are a generalization of IC signatures that allow even passively-corrupted parties to reliably verify signatures even without an honest majority. We then use these group commitments to construct a verifiable secret-sharing scheme, and describe how to perform computations on shared values.

Group Commitments. As a first step, we introduce the notion of *group commitments*, which is a pair of protocols GROUPCOMMIT and GROUPREVEAL. GROUPCOMMIT allows a group $\mathcal{G}$ to commit to a value v on which they agree (while providing secrecy with respect to the remaining parties $\mathcal{P} \setminus \mathcal{G}$), and GROUPREVEAL allows them to reveal this value to the remaining parties. Our definitions and protocols for group commitments are based on the IC signatures introduced in Sec. 2.

Definition 3 (IC Group Commitment). *A group $\mathcal{G}$ is* IC group committed *(or simply* committed*) to a value v, denoted by $\langle\!\langle v \rangle\!\rangle_{\mathcal{G}}$, if for all pairs $(p_i, p_j) \in \mathcal{G} \times \mathcal{G}$, v is IC-signed with $\langle v \rangle_{i,j}$.*

Note that a default group commitment $\langle\!\langle v \rangle\!\rangle_{\mathcal{G}}$ can be generated given that all parties in $\mathcal{P}$ know the value v. Furthermore, if all parties in $\mathcal{G}$ are actively corrupted, then any values held by correct parties constitute a valid group commitment. Additionally, group commitments inherit linearity from the underlying IC signature scheme.

Protocol GROUPCOMMIT: Given a set $\mathcal{G}$ of parties that agree on a value v, compute a valid group commitment $\langle\!\langle v \rangle\!\rangle_{\mathcal{G}}$ on v.

1. For each pair $(p_i, p_j) \in \mathcal{G} \times \mathcal{G}$ invoke IC-SIGN on v with signer p_i and intermediary p_j.
2. If any invocation of IC-SIGN outputs $\perp$, all parties output $\perp$. Otherwise, each party outputs the concatenation of the outputs of the invocations of IC-SIGN.

Fig. 3. The group commit protocol for a group $\mathcal{G}$.

Lemma 3. *Given a set $\mathcal{G}$ of parties that agree on a value v. If all parties in $\mathcal{G}$ are correct (i.e. $\mathcal{G} \cap \mathcal{D}^* = \emptyset$),* GROUPCOMMIT *correctly computes a valid group commitment $\langle\!\langle v \rangle\!\rangle_{\mathcal{G}}$ on v. Otherwise,* GROUPCOMMIT *either correctly computes a valid group commitment $\langle\!\langle v \rangle\!\rangle_{\mathcal{G}}$ on v, or all parties in $\mathcal{P}$ output $\perp$.* GROUPCOMMIT *is always secret and robust.*

Proof. SECRECY and ROBUSTNESS follow immediately by inspection. For CORRECTNESS, we first have to show that if the protocol outputs a group commitment, then all signatures held by correct parties p_j are for the value v. This follows immediately from the fact that IC-SIGN always results either in a correct signature $\langle v \rangle_{i,j}$ or in $\perp$, even when the signer (or intermediary) is actively corrupted. Second, if all parties in $\mathcal{G}$ are correct, then it follows from the properties of IC-SIGN that it never outputs $\perp$. □

If a group $\mathcal{G}$ is committed to a value v (e.g. if the GROUPCOMMIT protocol resulted in a valid group commitment and did not output $\perp$), the GROUPREVEAL protocol reveals the value v to all parties in $\mathcal{P}$. During the protocol run, the adversary might be able to provoke conflicts that depend on the sets $\mathcal{D}^*$ and $\mathcal{E}^*$ of corrupted parties. Therefore, we introduce a parameter $\mathcal{R}$, which is a monotone collection of pairs $(\mathcal{D}, \mathcal{E})$ of subsets of the player set, where $\mathcal{D} \subseteq \mathcal{E}$: Whenever all conflicts in a given situation can be explained with an adversary $(\mathcal{D}, \mathcal{E}) \in \mathcal{R}$, the corresponding values are ignored (corrected), and the protocol proceeds; otherwise it aborts. Note that GROUPREVEAL is the only subprotocol that might abort. All other protocols abort only if they use GROUPREVEAL as a subprotocol. Therefore, it is sufficient to discuss agreement on abort only for this protocol.

We emphasize that the conflicts in GROUPREVEAL do not only depend on the set $\mathcal{D}^*$ of actively corrupted parties, but also on the set $\mathcal{E}^*$ of passively corrupted parties, due to their inability to reliably verify IC-signatures. That means, in this protocol, even passive corruptions have a strong impact on correctness (and robustness).

Protocol GROUPREVEAL: Given the set $\mathcal{G}$ and a group commitment $\langle\!\langle v \rangle\!\rangle_{\mathcal{G}}$, reveal v to all parties.

1. For each party $p_i \in \mathcal{G}$:
   (a) p_i broadcasts v. Denote the broadcasted value with u_i.
   (b) For each party $p_j \in \mathcal{G}$: Invoke IC-REVEAL on $\langle v \rangle_{j,i}$.
   (c) A party $p_k \in \mathcal{P} \setminus \mathcal{G}$ accepts u_i if all invocations of IC-REVEAL output ("accept", u_i).
2. For each party $p_k \in \mathcal{P} \setminus \mathcal{G}$:
   (a) If p_k accepted at least one value in Step 1(c), and all accepted values are the same, then set u_k to this value. Else set $u_k := \perp$.
   (b) p_k broadcasts u_k.
3. Let $\mathcal{V}_u$ denote the set of parties that broadcasted u in Step 1(a) of 2(b), respectively. If $\exists (\mathcal{D}, \mathcal{E}) \in \mathcal{R}$ and a value v', such that $\mathcal{P} \setminus (\mathcal{V}_\perp \cup \mathcal{V}_{v'}) \subseteq \mathcal{D} \ \wedge \ \big(\mathcal{G} \subseteq \mathcal{E} \ \vee \ \mathcal{P} \setminus \mathcal{V}_{v'} \subseteq \mathcal{E}\big)$ then output v'. Else abort.

Fig. 4. The group reveal protocol for a group $\mathcal{G}$.

Lemma 4. *Given the reconstruction parameter $\mathcal{R}$, the commitment group $\mathcal{G}$, and a group commitment $\langle\!\langle v \rangle\!\rangle_{\mathcal{G}}$ for a value v,* GROUPREVEAL *reveals v to all parties. The protocol is statistically correct if $\mathcal{G} \not\subseteq \mathcal{D}^*$ and*

$$\forall (\mathcal{D}, \mathcal{E}) \in \mathcal{R} : \ \mathcal{G} \setminus \mathcal{D} \not\subseteq \mathcal{D}^* \ \vee \ (\mathcal{G} \not\subseteq \mathcal{E} \ \wedge \ \mathcal{P} \setminus \mathcal{E} \not\subseteq \mathcal{D}^*) \ \vee \ (\mathcal{G} \not\subseteq \mathcal{E}^* \ \wedge \ \mathcal{P} \setminus \mathcal{E}^* \not\subseteq \mathcal{D}).$$

The protocol is statistically robust if additionally $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{R}$, and always guarantees agreement on abort.

Proof. CORRECTNESS: Consider an actual protocol execution with correct value v and an adversary corrupting $(\mathcal{D}^*, \mathcal{E}^*)$. Denote with $\{\mathcal{V}_u\}$ the resulting collection of subsets of $\mathcal{P}$ in Step 3.

We first show that given the precondition $\mathcal{G} \not\subseteq \mathcal{D}^*$, we have

$$\big(\mathcal{P} \setminus (\mathcal{V}_\perp \cup \mathcal{V}_v) \subseteq \mathcal{D}^*\big) \ \wedge \ \big(\mathcal{G} \subseteq \mathcal{E}^* \ \vee \ \mathcal{P} \setminus \mathcal{V}_v \subseteq \mathcal{E}^*\big).$$

The precondition $\mathcal{G} \not\subseteq \mathcal{D}^*$ implies that there is at least one correct party $p_i \in \mathcal{G}$. In Step 1, this p_i broadcasts its value $u_i (= v)$ and invokes IC-REVEAL on the signatures $\langle v \rangle_{j,i}$ for $p_j \in \mathcal{G}$. It follows from the properties of IC-REVEAL that all correct parties accept all these signatures. Hence, all correct parties in $\mathcal{P} \setminus \mathcal{G}$ accept the value $u_i (= v)$, and broadcast either v or $\perp$ in Step 2, but not a wrong value, i.e. $\mathcal{P} \setminus (\mathcal{V}_\perp \cup \mathcal{V}_v) \subseteq \mathcal{D}^*$. Furthermore, either $\mathcal{G} \subseteq \mathcal{E}^*$, or there is an honest party $p_j \in \mathcal{G}$. In the latter case, an actively corrupted $p_i \in \mathcal{G}$ can only forge the signatures $\langle v \rangle_{j,i}$ towards passively corrupted parties. Hence, it is guaranteed that all honest parties p_k broadcast the correct value $u_k = v$ in Step 2, and we have $\mathcal{P} \setminus \mathcal{V}_v \subseteq \mathcal{E}^*$.

Second, we show that given the precondition in the lemma, the protocol execution under consideration does not output an (incorrect) value $v' \neq v$, i.e., for all $v' \neq v$ and $(\mathcal{D}, \mathcal{E}) \in \mathcal{R}$ the condition in Step 3 is violated. To arrive at a contradiction, assume that for some $v' \neq v$ and $(\mathcal{D}, \mathcal{E}) \in \mathcal{R}$ it holds that

$$\big(\mathcal{P} \setminus (\mathcal{V}_\perp \cup \mathcal{V}_{v'}) \subseteq \mathcal{D}\big) \ \wedge \ \big(\mathcal{G} \subseteq \mathcal{E} \ \vee \ \mathcal{P} \setminus \mathcal{V}_{v'} \subseteq \mathcal{E}\big). \tag{I}$$

From above, we have that

$$\big(\mathcal{P} \setminus (\mathcal{V}_\perp \cup \mathcal{V}_v) \subseteq \mathcal{D}^*\big) \ \wedge \ \big(\mathcal{G} \subseteq \mathcal{E}^* \ \vee \ \mathcal{P} \setminus \mathcal{V}_v \subseteq \mathcal{E}^*\big). \tag{II}$$

Furthermore, by assumption we have that the precondition in the lemma is fulfilled. We split the proof according to which or-term of the second part of this precondition is fulfilled for the given $(\mathcal{D}, \mathcal{E})$:

Case $\mathcal{G} \setminus \mathcal{D} \not\subseteq \mathcal{D}^*$: Since $\mathcal{P} \setminus (\mathcal{V}_\perp \cup \mathcal{V}_{v'}) \subseteq \mathcal{D}$ (I) and $\mathcal{G} \subseteq \mathcal{P}$, we have $\mathcal{G} \setminus (\mathcal{V}_\perp \cup \mathcal{V}_{v'}) \subseteq \mathcal{D}$. It follows by inspection of the protocol that $\mathcal{G}$ and $\mathcal{V}_\perp$ are disjoint. Hence we have $\mathcal{G} \setminus \mathcal{V}_{v'} \subseteq \mathcal{D}$. Analogously, it follows from $\mathcal{P} \setminus (\mathcal{V}_\perp \cup \mathcal{V}_v) \subseteq \mathcal{D}^*$ (II) that $\mathcal{G} \setminus \mathcal{V}_v \subseteq \mathcal{D}^*$. Therefore we have that $\mathcal{G} \subseteq \mathcal{D} \cup \mathcal{D}^*$, which is a contradiction to $\mathcal{G} \setminus \mathcal{D} \not\subseteq \mathcal{D}^*$.

Case $\mathcal{G} \not\subseteq \mathcal{E} \ \wedge \ \mathcal{P} \setminus \mathcal{E} \not\subseteq \mathcal{D}^*$**:** Since $\mathcal{G} \not\subseteq \mathcal{E}$, we have $\mathcal{P} \setminus \mathcal{V}_{v'} \subseteq \mathcal{E}$ (I). Furthermore, we have that $\mathcal{P} \setminus (\mathcal{V}_\perp \cup \mathcal{V}_v) \subseteq \mathcal{D}^*$ (II). It follows by inspection from the protocol that $\mathcal{V}_\perp$, $\mathcal{V}_{v'}$, and $\mathcal{V}_v$ are pairwise disjoint. Hence, we have that $\mathcal{P} \subseteq \mathcal{D}^* \cup \mathcal{E}$, which is a contradiction to $\mathcal{P} \setminus \mathcal{E} \not\subseteq \mathcal{D}^*$.

Case $\mathcal{G} \not\subseteq \mathcal{E}^* \ \wedge \ \mathcal{P} \setminus \mathcal{E}^* \not\subseteq \mathcal{D}$**:** This proof is identical to the previous case, with the only difference that $(\mathcal{D}^*, \mathcal{E}^*)$ is swapped with $(\mathcal{D}, \mathcal{E})$ and v with v'.

ROBUSTNESS: In the proof of correctness, we have shown that

$$\big(\mathcal{P} \setminus (\mathcal{V}_\perp \cup \mathcal{V}_v) \subseteq \mathcal{D}^*\big) \ \wedge \ \big(\mathcal{G} \subseteq \mathcal{E}^* \ \vee \ \mathcal{P} \setminus \mathcal{V}_v \subseteq \mathcal{E}^*\big).$$

Hence, given the correctness condition and $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{R}$, it follows immediately that the condition in Step 3 is fulfilled for the correct value v and $(\mathcal{D}^*, \mathcal{E}^*)$, i.e., that the protocol terminates without abort.

AGREEMENT ON ABORT: Since the abort decision is based only on broadcasted values, we always have agreement on abort. □

Verifiable Secret Sharing. The state of the protocol is maintained with a sum-sharing, where each party holds several summands. Furthermore, for each summand s_i, the group of those parties that hold s_i is committed to it.

Definition 4 ($\mathcal{S}$-Sharing). *A value s is* $\mathcal{S}$-shared *for sharing specification $\mathcal{S} = (S_1, \ldots, S_\ell)$ if (1) there are values $s_1, \ldots, s_\ell$, such that $s_1 + \ldots + s_\ell = s$, (2) for all i, every (correct) party $p_j \in S_i$ holds the summand s_i, and (3) each group S_i is committed to s_i with a group commitment $\langle\!\langle s_i \rangle\!\rangle_{S_i}$. A sharing specification $\mathcal{S}$ is* $\mathcal{D}$-permissive*, if each summand is held by at least one party outside $\mathcal{D}$, i.e. $\forall i : S_i \setminus \mathcal{D} \neq \emptyset$.*

Lemma 5. *Let $\mathcal{S}$ be the sharing specification. An $\mathcal{S}$-sharing is secret if $\exists S_i \in \mathcal{S} : S_i \cap \mathcal{E}^* = \emptyset$, and uniquely defines a value if $\mathcal{S}$ is $\mathcal{D}^*$-permissive.*

Proof. Secrecy follows from the fact that $\mathcal{E}^*$ lacks at least one summand s_i. Furthermore, given that $\mathcal{S}$ is $\mathcal{D}^*$-permissive, each summand s_i is held by at least one correct party. Hence, the secret s is uniquely defined by $s = s_1 + \ldots + s_\ell$. □

The share protocol takes as input a secret s from a dealer, and outputs an $\mathcal{S}$-sharing of s (see Fig. 5).

Protocol SHARE$^{\mathrm{GA}}$: Given input s from the dealer, compute an $\mathcal{S}$-sharing of this value.

1. The dealer chooses uniformly random summands $s_1, \ldots, s_\ell$ with $\sum_{i=1}^{\ell} s_i = s$, where $\ell = |\mathcal{S}|$. Then, for each $S_i \in \mathcal{S}$, the dealer sends s_i to every party $p_j \in S_i$.
2. For all $S_i \in \mathcal{S}$: Every party $p_j \in S_i$ sends s_i to every other party in S_i. Then, every party in S_i broadcasts a complaint bit, indicating whether it observed an inconsistency.
3. For all $S_i \in \mathcal{S}$, for which no inconsistency was reported, GROUPCOMMIT is invoked to compute $\langle\!\langle s_i \rangle\!\rangle_{S_i}$.
4. The dealer broadcasts each summand s_i for which either an inconsistency was reported (Step 2), or the output of GROUPCOMMIT was $\perp$ (Step 3). The players in S_i accept this summand, and a default group commitment is used. If the dealer does not broadcast a summand s_i, the parties use $s_i = 0$ with a default group commitment.
5. Each party p_j outputs its share $\{s_i \mid p_j \in S_i\}$ together with its part of the group commitments.

Fig. 5. The share protocol for general adversaries.

Lemma 6. *Let $\mathcal{S}$ be the sharing specification. On input s from the dealer,* SHARE$^{\mathrm{GA}}$ *correctly, secretly and robustly computes an $\mathcal{S}$-sharing. If $\mathcal{S}$ is $\mathcal{D}^*$-permissive, and if the dealer is correct, the sharing uniquely defines the secret s.*

Proof. SECRECY: Given a correct dealer, the summands distributed in the first step are consistent. In the remaining protocol run, no additional information is revealed to the adversary: A summand s_i is broadcasted only if a party $p_j \in S_i$ reported an inconsistency, or GROUPCOMMIT outputs $\perp$. Yet, this occurs only if one of the parties in S_i is actively corrupted, i.e., when the adversary knew s_i already beforehand. Furthermore, it follows from the properties of GROUPCOMMIT that secrecy is maintained during its invocations.
CORRECTNESS: First, we have to show that the protocol outputs a valid $\mathcal{S}$-sharing. Due to the bilateral checks, any inconsistency in the summands held by correct parties is detected in Step 2 and resolved in Step 4. Furthermore, it follows from the properties of GROUPCOMMIT that in Step 3, either a correct group commitment is computed, or all parties output $\perp$. In the latter case, a default (and hence correct) group commitment is used (Step 4). Therefore, the output is a valid $\mathcal{S}$-sharing. Second, we have to show that if $\mathcal{S}$ is $\mathcal{D}^*$-permissive and if the dealer is correct, then the shared value equals the input of the dealer. A correct dealer always responds on reported inconsistencies with the original summands. Hence, the unique value defined by the sharing is the secret s.
ROBUSTNESS: It follows from inspection that the protocol does not abort. □

For the public reconstruction[7] of a shared value (Fig. 6), we use the fact that there is a group commitment for each summand of the sharing. These commitments allow to reliably reveal each summand using GROUPREVEAL.

Protocol PUBLIC RECONSTRUCTION$^{\text{GA}}$: Given an $\mathcal{S}$-sharing of some value s, reconstruct s to all parties.

1. For each summand s_i, invoke GROUPREVEAL on $\langle\!\langle s_i \rangle\!\rangle_{S_i}$.
2. Each party outputs the secret $s = s_1 + \ldots + s_\ell$.

Fig. 6. The public reconstruction protocol for general adversaries.

Lemma 7. *Given the sharing specification $\mathcal{S}$, the robustness parameter $\mathcal{R}$, and an $\mathcal{S}$-sharing of some value s,* PUBLIC RECONSTRUCTION$^{\text{GA}}$ *reconstructs s to all parties. The protocol is correct if*

$$\forall (\mathcal{D}, \mathcal{E}) \in \mathcal{R}, S \in \mathcal{S}: \quad S \not\subseteq \mathcal{D}^* \quad \wedge$$
$$\big(S \setminus \mathcal{D} \not\subseteq \mathcal{D}^* \ \vee\ (S \not\subseteq \mathcal{E} \ \wedge\ \mathcal{P} \setminus \mathcal{E} \not\subseteq \mathcal{D}^*) \ \vee\ (S \not\subseteq \mathcal{E}^* \ \wedge\ \mathcal{P} \setminus \mathcal{E}^* \not\subseteq \mathcal{D})\big)$$

and robust if additionally $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{R}$.

Proof. Given the condition for correctness in the lemma, all invocations of GROUPREVEAL are correct. The same holds for robustness. Then, all security properties follow directly from the security of GROUPREVEAL.

Addition, Multiplication, and Random Values. Linear functions (and in particular additions) can be computed locally, since $\mathcal{S}$-sharings and group commitments are linear. In particular, given sharings of a and b, and a constant c, one can easily compute sharings of $a + b$, ca, and $a + c$. Computing a shared random value can be achieved by letting each party p_i share a random value r_i, and computing a sharing of $r = r_1 + \ldots + r_n$.

For the multiplication of two values a and b, we adapt the protocol from [HMZ08] by using our modified share and reconstruct protocols (Fig. 7). The multiplication protocol exploits the fact that $ab = \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} a_i b_j$: For each $a_i b_j$, one party that knows a_i and b_j computes $v_{ij} = a_i b_j$, shares it, and proves that the sharing contains the correct value. Then, the parties compute the linear function described above on these sharings.

[7] Private reconstruction can easily be reduced to public reconstruction [CDG88, HLMR11].

In order to prove that the sharing contains the correct value, the corresponding party provides a zero-knowledge proof by invoking ABC-PROOF$^{\mathrm{GA}}$ on sharings of a_i, b_i, and v_{ij}. If this proof is not accepted, this party (the prover) is actively corrupted, and the summands a_i and b_j can be reconstructed without violating secrecy. This zero-knowledge proof requires that a_i and b_j are $\mathcal{S}$-shared, which we achieve by invoking GROUPSHARE$^{\mathrm{GA}}$, a subprotocol that allows to share individual summands.

Protocol MULTIPLICATION$^{\mathrm{GA}}$: Given $[a]$ and $[b]$, compute $[c]$ for $c = ab$.

1. For each pair $S_i, S_j \in \mathcal{S}$, let p_{ij} denote the party with the smallest index in $S_i \cap S_j$.
   (a) p_{ij} computes $v_{ij} = a_i b_j$ and invokes SHARE$^{\mathrm{GA}}$ on it, resulting in $[v_{ij}]$.
   (b) Invoke GROUPSHARE$^{\mathrm{GA}}$ on $\langle\!\langle a_i \rangle\!\rangle_{S_i}$ and $\langle\!\langle b_j \rangle\!\rangle_{S_j}$ both with dealer p_{ij}, resulting in $[a_i]$ and $[b_j]$.
   (c) Invoke ABC-PROOF$^{\mathrm{GA}}$ on $[a_i]$, $[b_j]$, and $[v_{ij}]$ with prover p_{ij}. If the proof is rejected, invoke PUBLIC RECONSTRUCTION$^{\mathrm{GA}}$ on $[a_i]$ and $[b_j]$, and use a default $\mathcal{S}$-sharing of $v_{ij} := a_i b_j$.
2. The parties (distributively) compute the sum of all sharings $[v_{ij}]$, resulting in a sharing of $c = ab$.

Fig. 7. The multiplication protocol for general adversaries.

Lemma 8. *Given the sharing specification $\mathcal{S}$, the robustness parameter $\mathcal{R}$, and $\mathcal{S}$-sharings of a and b,* MULTIPLICATION$^{\mathrm{GA}}$ *computes an $\mathcal{S}$-sharing of the product $c = ab$. The protocol is correct if* $\forall S, S' \in \mathcal{S} : S \cap S' \neq \emptyset$ *and*

$$\forall (\mathcal{D}, \mathcal{E}) \in \mathcal{R}, S \in \mathcal{S} : \quad S \not\subseteq \mathcal{D}^* \quad \wedge$$
$$\big(S \setminus \mathcal{D} \not\subseteq \mathcal{D}^* \quad \vee \quad (S \not\subseteq \mathcal{E} \ \wedge \ \mathcal{P} \setminus \mathcal{E} \not\subseteq \mathcal{D}^*) \quad \vee \quad (S \not\subseteq \mathcal{E}^* \ \wedge \ \mathcal{P} \setminus \mathcal{E}^* \not\subseteq \mathcal{D})\big),$$

robust if additionally $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{R}$, and always secret.

Proof. The condition $\forall S, S' \in \mathcal{S} : S \cap S' \neq \emptyset$ implies that every value $a_i b_j$ can be computed by at least one party. Furthermore, given the condition for correctness in the lemma, all subprotocols are correct. The same holds for robustness. All subprotocols are always secret. Given these observations, it follows by inspection that the protocol is secure. □

The GROUPSHARE$^{\mathrm{GA}}$ subprotocol (Fig. 8) allows to reshare summands: Given a group S_i of parties that is committed to a value s_i with a group commitment $\langle\!\langle s_i \rangle\!\rangle_{S_i}$ and a dealer $p \in S_i$, the protocol GROUPSHARE$^{\mathrm{GA}}$ computes an $\mathcal{S}$-sharing $[s_i]$ of the value s_i, given that at least one party in S_i is correct.

Protocol GROUPSHARE$^{\mathrm{GA}}$: Given a group S_i, a group commitment $\langle\!\langle s_i \rangle\!\rangle_{S_i}$, and a dealer $p \in S_i$, compute $[s_i]$.

1. p invokes SHARE$^{\mathrm{GA}}$ on s_i, resulting in $[s_i]$.
2. For each $p_j \in S_i$: Invoke PRIVATE RECONSTRUCTION$^{\mathrm{GA}}$ on $[s_i]$ for p_j. If the reconstructed value is not s_i, p_j broadcasts "reject". Otherwise, it broadcasts "accept".
3. If some $p_j \in S_i$ broadcasted "reject", invoke GROUPREVEAL on $\langle\!\langle s_i \rangle\!\rangle_{S_i}$, and use a default sharing of s_i.
4. Each party outputs its share of $[s_i]$, and p outputs the vector of summands of $[s_i]$.

Fig. 8. The group share protocol for general adversaries.

Lemma 9. *Given the sharing specification $\mathcal{S}$, the reconstruction parameter $\mathcal{R}$, a group $S_i \in \mathcal{S}$, a group commitment $\langle\!\langle s_i \rangle\!\rangle_{S_i}$, and a dealer $p \in S_i$,* GROUPSHARE$^{\mathrm{GA}}$ *computes $[s_i]$. The protocol is correct if $S_i \setminus \mathcal{D}^* \neq \emptyset$ and the subprotocols are correct against $(\mathcal{D}^*, \mathcal{E}^*)$. Furthermore, it guarantees secrecy and/or robustness against $(\mathcal{D}^*, \mathcal{E}^*)$ whenever the subprotocols provide the corresponding guarantee against $(\mathcal{D}^*, \mathcal{E}^*)$.*

Proof. CORRECTNESS: Since $S_i \setminus \mathcal{D}^* \neq \emptyset$, there is at least one correct party that observes any inconsistency and complains. Hence, the shared value is correct.
SECRECY: In Step 3, s_i is publicly reconstructed only if either p or some $p_j \in S_i$ is actively corrupted. Hence, the adversary knew the value already before.
ROBUSTNESS: It follows from inspection that the protocol does not abort. □

Next, we present a subprotocol that allows a prover p to prove that a given sharing $[c]$ contains the product of the values of two other given sharings $[a]$ and $[b]$ (Fig. 9). The protocol is along the lines of the protocol in [CDD+99].

Protocol ABC-PROOF$^{\mathrm{GA}}$: Given $[a]$, $[b]$, and $[c]$, where prover p knows a, b, and c, check whether $c = ab$.

1. For each party $p_j \in \mathcal{P}$, carry out a sub-proof:
   (a) p chooses a uniformly random b', computes $c' = ab'$, and invokes SHARE$^{\mathrm{GA}}$ on both b' and c', resulting in $[b']$ and $[c']$.
   (b) p_j broadcasts a uniformly random challenge r.
   (c) Compute $[b''] = r[b] + [b']$ and invoke PUBLIC RECONSTRUCTION$^{\mathrm{GA}}$ on $[b'']$.
   (d) Compute $[z] = b''[a] - r[c] - [c']$ and invoke PUBLIC RECONSTRUCTION$^{\mathrm{GA}}$ on $[z]$.
   (e) If $z = 0$ the sub-proof is accepted. Otherwise, it is rejected.
2. If any of the sub-proofs was rejected, output "reject". Otherwise output "accept".

Fig. 9. The protocol for proving that $c = ab$ for general adversaries.

Lemma 10. *Given are the sharing specification $\mathcal{S}$, the reconstruction parameter $\mathcal{R}$, and $\mathcal{S}$-sharings $[a]$, $[b]$, and $[c]$, where prover p knows a, b, and c. Assume that the subprotocols are correct against $(\mathcal{D}^*, \mathcal{E}^*)$. If p is correct and $c = ab$, then* ABC-PROOF$^{\mathrm{GA}}$ *outputs "accept". If $c \neq ab$, then* ABC-PROOF$^{\mathrm{GA}}$ *outputs "reject" with overwhelming probability. Furthermore, the protocol guarantees secrecy and/or robustness against $(\mathcal{D}^*, \mathcal{E}^*)$ whenever the subprotocols provide the corresponding guarantee against $(\mathcal{D}^*, \mathcal{E}^*)$.*

Proof. CORRECTNESS: If the dealer is correct and $c = ab$, then it follows by simple arithmetic that all sub-proofs are accepted. It remains to show that if $c \neq ab$, then at least one sub-proof is rejected with overwhelming probability. We first show that if $z = 0$ for any two challenges r and r' where $r \neq r'$, then we must have $c = ab$: If $z = 0$ for r and r', then $a(rb + b') - cr - c' = a(r'b + b') - cr' - c'$. This can be written as $ab(r - r') = c(r - r')$. Since, $r \neq r'$, it follows that $c = ab$. Hence, if $c \neq ab$, then the sub-proof is accepted for at most one challenge. Since an actively corrupted prover does not know the challenges from correct parties in advance, an incorrect c is detected with overwhelming probability.
SECRECY: The only values revealed during the protocol are b'' and z. If p is correct, then b'' is perfectly blinded by b', and $z = 0$.
ROBUSTNESS: It follows from inspection that the protocol does not abort. □

The Security of the Parametrized Protocol. Considering the security of the subprotocols described above, we can derive the security of the parametrized protocol, denoted by $\pi^{\mathcal{S},\mathcal{R}}$:

Lemma 11. *Given the sharing specification $\mathcal{S}$ and the reconstruction parameter $\mathcal{R}$, the protocol $\pi^{\mathcal{S},\mathcal{R}}$ guarantees* statistical correctness *if*

$$\forall (\mathcal{D}, \mathcal{E}) \in \mathcal{R}, S, S' \in \mathcal{S}: \quad S \cap S' \neq \emptyset \quad \wedge \quad S \not\subseteq \mathcal{D}^* \quad \wedge \\ \big(S \setminus \mathcal{D} \not\subseteq \mathcal{D}^* \ \vee \ (S \not\subseteq \mathcal{E} \ \wedge \ \mathcal{P} \setminus \mathcal{E} \not\subseteq \mathcal{D}^*) \ \vee \ (S \not\subseteq \mathcal{E}^* \ \wedge \ \mathcal{P} \setminus \mathcal{E}^* \not\subseteq \mathcal{D})\big)$$

Furthermore, the protocol guarantees statistical secrecy *if additionally $\exists S \in \mathcal{S} : S \cap \mathcal{E}^* = \emptyset$, and/or* statistical robustness *if additionally $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{R}$.*

Proof. $\pi^{\mathcal{S},\mathcal{R}}$ provides a certain security guarantee against $(\mathcal{D}^*, \mathcal{E}^*)$ if all subprotocols (cf. Lemmas 3 to 10) and the sharing (cf. Lemma 5) provide this guarantee against $(\mathcal{D}^*, \mathcal{E}^*)$. For each guarantee, it can easily be verified that the condition in the lemma implies the conditions in the corresponding lemmas. □

3.2 Main Result

The following theorem states the optimal bound for statistically secure MPC for general adversaries with both mixed adversaries and hybrid security. We show that the bound is sufficient for MPC by providing parameters for the generalized protocols described above. In the next section, we prove that the bound is also necessary.

Theorem 1. *In the secure channels model with broadcast and general adversaries, statistically secure (reactive) MPC among $n \geq 2$ parties with respect to $(\mathcal{Z}^c, \mathcal{Z}^s, \mathcal{Z}^r, \mathcal{Z}^f)$, where $\mathcal{Z}^r \subseteq \mathcal{Z}^c$ and $\mathcal{Z}^f \subseteq \mathcal{Z}^s \subseteq \mathcal{Z}^c$, is possible if*

$$\begin{aligned}
&\mathcal{Z}^s = \{(\emptyset,\emptyset)\} \quad \vee \\
&\forall (\cdot,\mathcal{E}^s), (\cdot,\mathcal{E}^{s\prime}) \in \mathcal{Z}^s, (\mathcal{D}^r,\mathcal{E}^r) \in \mathcal{Z}^r, (\mathcal{D}^c,\mathcal{E}^c) \in \mathcal{Z}^c \ : \\
&\quad \mathcal{E}^s \cup \mathcal{E}^{s\prime} \neq \mathcal{P} \quad \wedge \quad \mathcal{E}^s \cup \mathcal{D}^c \neq \mathcal{P} \quad \wedge \\
&\quad \Big(\mathcal{D}^c \cup \mathcal{D}^r \cup \mathcal{E}^s \neq \mathcal{P} \ \vee \ (\mathcal{E}^s \cup \mathcal{E}^r \neq \mathcal{P} \wedge \mathcal{D}^c \cup \mathcal{E}^r \neq \mathcal{P}) \ \vee \ (\mathcal{E}^s \cup \mathcal{E}^c \neq \mathcal{P} \wedge \mathcal{D}^r \cup \mathcal{E}^c \neq \mathcal{P})\Big)
\end{aligned}$$

This bound is tight: If violated, there are (reactive) functionalities that cannot be securely computed.

Proof (Sufficiency). If $\mathcal{Z}^s = \{(\emptyset,\emptyset)\}$, there is no secrecy requirement, and we can directly use the trivial non-secret protocol described in the Appendix of [HLMR11]. Otherwise, we employ the protocol $\pi^{\mathcal{S},\mathcal{R}}$ described in Sec. 3.1. We set $\mathcal{S} := \{\overline{\mathcal{E}^s} \mid (\cdot,\mathcal{E}^s) \in \mathcal{Z}^s\}$ and $\mathcal{R} = \mathcal{Z}^r \cup \mathcal{Z}^f$.

We apply Lemma 11 to derive correctness, secrecy and robustness: Given the bound in the theorem, the choice of the structures $\mathcal{S}$ and $\mathcal{R}$, and the fact that $(\mathcal{D}^*, \mathcal{E}^*)$ is an element of the corresponding adversary structure, it is easy to verify that the condition for each property is fulfilled. In particular, note that the correctness condition is also fulfilled for $(\mathcal{D}, \mathcal{E}) \in \mathcal{Z}^f$: Using that $\mathcal{Z}^f \subseteq \mathcal{Z}^s$, we have that $\mathcal{E}^s \cup \mathcal{E} \subseteq \mathcal{E}^s \cup \mathcal{E}^{s\prime} \neq \mathcal{P}$ (for some $\mathcal{E}^{s\prime}$) and $\mathcal{D}^c \cup \mathcal{E} \subseteq \mathcal{D}^c \cup \mathcal{E}^s \neq \mathcal{P}$ (where the inequalities follow from the second line of the condition in the theorem). This implies the condition for correctness.

Note that by our choice of $\mathcal{R}$, we have $\mathcal{Z}^f \subseteq \mathcal{R}$. Hence, for $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^f$ the protocol is robust, and the adversary cannot abort. □

3.3 Proofs of Necessity

In this section, we prove that the bounds in Theorems 1 and 2 are necessary, i.e. if violated, MPC is impossible.[8] The bound in Thm. 1 is violated if

$$\begin{aligned}
&\mathcal{Z}^s \neq \{(\emptyset,\emptyset)\} \quad \wedge \\
&\quad \exists (\cdot,\mathcal{E}^s), (\cdot,\mathcal{E}^{s\prime}) \in \mathcal{Z}^s \ : \mathcal{E}^s \cup \mathcal{E}^{s\prime} = \mathcal{P} && (1) \\
&\vee \ \exists (\cdot,\mathcal{E}^s) \in \mathcal{Z}^s, (\mathcal{D}^c,\cdot) \in \mathcal{Z}^c \ : \mathcal{E}^s \cup \mathcal{D}^c = \mathcal{P} && (2) \\
&\vee \ \exists (\cdot,\mathcal{E}^s) \in \mathcal{Z}^s, (\mathcal{D}^r,\mathcal{E}^r) \in \mathcal{Z}^r, (\mathcal{D}^c,\mathcal{E}^c) \in \mathcal{Z}^c \ : && (3) \\
&\qquad \mathcal{D}^c \cup \mathcal{D}^r \cup \mathcal{E}^s = \mathcal{P} \ \wedge \ (\mathcal{E}^s \cup \mathcal{E}^r = \mathcal{P} \ \vee \ \mathcal{D}^c \cup \mathcal{E}^r = \mathcal{P}) \ \wedge \ (\mathcal{E}^s \cup \mathcal{E}^c = \mathcal{P} \ \vee \ \mathcal{D}^r \cup \mathcal{E}^c = \mathcal{P})
\end{aligned}$$

We split this condition according to which OR-term is fulfilled:

Case (1)**:** Assume that $\mathcal{Z}^s \neq \{(\emptyset,\emptyset)\} \wedge \exists \mathcal{E}^s, \mathcal{E}^{s\prime} : \mathcal{E}^s \cup \mathcal{E}^{s\prime} = \mathcal{P}$. Due to monotonicity, we can assume that $\mathcal{E}^s$ and $\mathcal{E}^{s\prime}$ are disjoint and (since $n \geq 2$) non-empty. In this case, impossibility of MPC follows from [RB89, Kil00].

[8] Note that the impossibility holds even when agreement on abort is not required.

Case (2)**:** Assume that $\mathcal{Z}^s \neq \{(\emptyset,\emptyset)\} \wedge \exists \mathcal{E}^s, \mathcal{D}^c : \mathcal{E}^s \cup \mathcal{D}^c = \mathcal{P}$. Due to monotonicity, we can assume that $\mathcal{E}^s$ and $\mathcal{D}^c$ are disjoint. Furthermore, since $\mathcal{Z}^s \neq \{(\emptyset,\emptyset)\}$, we can assume that $\mathcal{E}^s$ is non-empty. If $\mathcal{D}^c$ is empty, we have $\mathcal{E}^s = \mathcal{P}$, which is covered by the previous case. Otherwise, impossibility of MPC can easily be derived from the impossibility of IT secure commitments: Trivially, the impossibility holds for $|\mathcal{D}^c| = |\mathcal{E}^s| = 1$. All other cases can be reduced to the 2-party case by having each of the two parties emulate the parties in $\mathcal{D}^c$ and $\mathcal{E}^s$, respectively.

Case (3)**:** Assume that $\mathcal{Z}^s \neq \{(\emptyset,\emptyset)\} \wedge \exists(\cdot,\mathcal{E}^s) \in \mathcal{Z}^s, (\mathcal{D}^r,\mathcal{E}^r) \in \mathcal{Z}^r, (\mathcal{D}^c,\mathcal{E}^c) \in \mathcal{Z}^c :$
$\mathcal{D}^c \cup \mathcal{D}^r \cup \mathcal{E}^s = \mathcal{P} \quad \wedge \quad (\mathcal{E}^s \cup \mathcal{E}^r = \mathcal{P} \vee \mathcal{D}^c \cup \mathcal{E}^r = \mathcal{P}) \quad \wedge \quad (\mathcal{E}^s \cup \mathcal{E}^c = \mathcal{P} \vee \mathcal{D}^r \cup \mathcal{E}^c = \mathcal{P})$. Due to monotonicity, we can assume that the sets $\mathcal{E}^s$, $\mathcal{D}^r$, and $\mathcal{D}^c$ are disjoint, that $\mathcal{E}^r = \mathcal{D}^r \cup \mathcal{D}^c$ or $\mathcal{E}^r = \mathcal{D}^r \cup \mathcal{E}^s$, and that $\mathcal{E}^c = \mathcal{D}^c \cup \mathcal{D}^r$ or $\mathcal{E}^c = \mathcal{D}^c \cup \mathcal{E}^s$. Furthermore, we can assume that all these sets are non-empty: Since $\mathcal{Z}^s \neq \{(\emptyset,\emptyset)\}$, we have $\mathcal{E}^s \neq \emptyset$. If either $\mathcal{D}^r = \emptyset$ or $\mathcal{D}^c = \emptyset$, we have a reduction to the commitment impossibility.
Now, impossibility of MPC can be derived from Lemma 12: This is straight-forward for $|\mathcal{E}^s| = |\mathcal{D}^c| = |\mathcal{D}^r| = 1$. All other cases can be reduced to the 3-party case by having each of the three parties emulate the parties in $\mathcal{E}^s$, $\mathcal{D}^r$, and $\mathcal{D}^c$, respectively.

Reactive functionalities must be able to generate and hold a secret state (typically, this is achieved using a secret-sharing scheme). We prove that it is impossible to generate a state in a specific 3-party setting. This proof is inspired by [BFH+08, HLMR11].

Definition 5 (State and State Generation). *A* state *for n parties $p_1, \ldots, p_n$ is a tuple $(s_1, \ldots, s_n)$ that defines a value $r \in \{0, 1, \perp\}$, where party p_i holds s_i. A protocol* SHARE *for* state generation *allows a dealer to generate a state for an input bit s. Protocol* SHARE *must achieve*

1. secrecy*: The corrupted parties obtain no information about the bit s. In particular, the state information held by corrupted parties contains no information about the bit s.*
2. correctness*: The resulting state uniquely defines a value r, where $r \in \{s, \perp\}$ if the dealer is honest.*
3. robustness*: The resulting state uniquely defines a value $r \in \{0, 1\}$.*

Lemma 12. *Given three parties A, B, and C. On input a bit s from dealer C, the parties cannot generate a state (a, b, c) that defines s providing the following guarantees:*

1. *Statistical secrecy in case of a passively corrupted A.*
2. *Statistical correctness and robustness in case of an actively corrupted B and either passively corrupted A or passively corrupted C.*
3. *Statistical correctness (without agreement on abort) in case of an actively corrupted C and either passively corrupted A or passively corrupted B.*

Proof. Denote by T_A the transcript observed by party A during SHARE, and let a, b, and c be the resulting state information held by A, B, and C respectively.

To arrive at a contradiction, assume that (a, b, c) is a state generated by SHARE on input $s = 0$ (i.e., due to completeness, it defines 0 with overwhelming probability). Due to secrecy in case of a passively corrupted A, for any a, with overwhelming probability, there exist b' and c' such that (a, b', c') is a state defining $s = 1$ with overwhelming probability. The state (a, b', c') occurs with (nearly, i.e. negligible distance) the same probability as (a, b, c) (otherwise, a would give information about s).

Due to correctness and robustness in presence of an actively corrupted B, the state $(a, \cdot, c)$ defines the value 0 with overwhelming probability (where $\cdot$ is a placeholder for an arbitrary state information held by B). Due to correctness in presence of an actively corrupted C, the state $(a, b', \cdot)$ defines either 1 or $\perp$ with overwhelming probability.

Consider the following attack by an adversary actively corrupting B and passively corrupting A or C: The adversary behaves honest during SHARE, with input $s = 0$. Denote the resulting state with (a, b, c). The adversary knows the transcript T_A of party A. As a consequence, he can compute b' and c' (with overwhelming probability), and achieve the state (a, b', c).

However, with the same probability, this state could have been achieved by an adversary actively corrupting C and passively corrupting A or B, mounting an analogous attack: Again, the adversary behaves honest during SHARE, but with input $s = 1$. Denote the resulting state with (a, b', c'). As in the previous case, the adversary knows the transcript T_A of party A. As a consequence, he can compute b and c (with overwhelming probability), and also achieve the state (a, b', c).

Hence, the state (a, b', c) must define both 0 and either 1 or $\perp$ with overwhelming probability, which is a contradiction. □

4 MPC with Threshold Adversaries

Trivially, the protocol for general adversaries can also be applied to the special case of threshold adversaries. Yet, protocols for general adversaries are superpolynomial in the number of parties for most adversary structures. Therefore, we present a protocol that exploits the symmetry of threshold adversaries, and is efficient in the number of parties.

The characterization for general adversaries (Sec. 3) can be adjusted for threshold adversaries: A mixed adversary is characterized by two thresholds (t_a, t_p), where he may corrupt up to t_p parties passively, and up to t_a of these parties even actively. The level of security (correctness, secrecy, robustness, and fairness) depends only on the number $(|\mathcal{D}^*|, |\mathcal{E}^*|)$ of actually corrupted parties. In the perfect setting [HLMR11], this is modeled with four pairs of thresholds, one for each security requirement, specifying the upper bound on the number of corruptions that the adversary may perform, such that the corresponding security requirement is still guaranteed. In the statistical setting, it follows from the bound for general adversaries that we need to consider multiple pairs of thresholds for each security guarantee. Consider the following example: Let $n = 6$ and $t_p^s = 2$. It is possible to obtain correctness for $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq (2, 6)$ and $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq (3, 3)$, and robustness for $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq (1, 6)$ and $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq (2, 3)$ in the same protocol. Yet, correctness and robustness cannot be guaranteed for $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq (3, 6)$ and $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq (2, 6)$, respectively. Hence, this situation cannot be captured using only a single pair of thresholds for each security guarantee. Therefore, we introduce multi-thresholds T, i.e. collections of pairs of thresholds (t_a, t_p).

We consider the four multi-thresholds T^c, T^s, T^r, and T^f:[9] Correctness is guaranteed for $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq T^c$,[10] secrecy is guaranteed for $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq T^s$, robustness is guaranteed for $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq T^r$, and fairness is guaranteed for $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq T^f$. Again, we have the assumption that $T^r \leq T^c$ and $T^f \leq T^s \leq T^r$,[11] as secrecy and robustness are not well defined without correctness, and as fairness cannot be achieved without secrecy.

For threshold adversaries, we proceed along the lines of the general adversary case: We generalize the protocol of [FHM98, CDD+99] and introduce the *sharing parameter* d (corresponding to $\mathcal{S}$), and the *reconstruction parameter* E (corresponding to $\mathcal{R}$). Since we consider multi-thresholds, the reconstruction parameter E is a list of pairs (e_a, e_p) where $e_a \leq e_p$. Since for secrecy the actively corrupted parties $\mathcal{D}^*$ are not relevant, there cannot be two incomparable maximal adversaries. Hence, a single threshold is sufficient.

In this section, we assume that each party p_i is assigned a unique and publicly known evaluation point $\alpha_i \in \mathbb{F} \setminus \{0\}$. This implies that the field $\mathbb{F}$ must have more than n elements.

[9] As in the setting with general adversaries, we do not introduce a separate multi-threshold for agreement on abort.

[10] We write $(t_a, t_p) \leq T$ if $\exists (t'_a, t'_p) \in T : (t_a, t_p) \leq (t'_a, t'_p)$, where $(t_a, t_p) \leq (t'_a, t'_p)$ is a shorthand for $t_a \leq t'_a$ and $t_p \leq t'_p$.

[11] We write $T_1 \leq T_2$ if $\forall (t_a, t_p) \in T_1, \exists (t'_a, t'_p) \in T_2 : (t_a, t_p) \leq (t'_a, t'_p)$.

4.1 A Parametrized Protocol for Threshold Adversaries

In the following, we present the parametrized subprotocols and analyze them with respect to correctness, secrecy, and robustness. The main result (including fairness) is discussed in Sec. 4.2. The protocol is based on IC signatures as introduced in Sec. 2.

Verifiable Secret Sharing. The state of the protocol is maintained with a Shamir sharing [Sha79] of each intermediate result.

Definition 6 (d-Sharing). *A value s is d-*shared *when (1) there is a polynomial $\hat{s}(x)$ of degree d with $\hat{s}(0) = s$, and every party p_i holds a share $s_i = \hat{s}(\alpha_i)$, (2) for each share s_i, p_i holds a share polynomial $\hat{s}_i(y)$ of degree d with $\hat{s}_i(0) = s_i$, and every party p_j holds a share share $s_{ij} = \hat{s}_i(\alpha_j)$, and (3) for each share share s_{ij}, party p_i holds a signature $\langle s_{ij} \rangle_{j,i}$, and p_j holds a signature $\langle s_{ij} \rangle_{i,j}$. We denote a d-sharing of s with $[s]$, and the share s_i with $[s]_i$. A sharing parameter d is t-*permissive, *if the shares of all but t parties uniquely define the secret, i.e., $n - t > d$.*

Note that it follows from the linearity of Shamir sharings (i.e. a polynomial $\hat{s}(x)$ with $\hat{s}(0) = s$ where each party $p_j \in \mathcal{P}$ holds $\hat{s}(\alpha_j)$) and IC signatures, that d-sharings are linear.

Lemma 13. *Let $d < n$ be the sharing parameter. A d-sharing is secret if $|\mathcal{E}^*| \leq d$, and uniquely defines a value if d is $|\mathcal{D}^*|$-permissive.*

Proof. It follows directly from the properties of a polynomial of degree d that secrecy is guaranteed if the number $|\mathcal{E}^*|$ of (actively or passively) corrupted parties is at most d. Furthermore, $n - |\mathcal{D}^*| > d$ implies that there are at least $d + 1$ correct parties whose shares uniquely define a share polynomial. □

The share protocol takes as input a secret s from a dealer, and outputs a d-sharing $[s]$ (see Fig. 10).

Protocol SHARE: Given input s from the dealer, compute a d-sharing $[s]$ of this value.

1. The dealer chooses a random (bivariate) polynomial $g(x, y)$ with $g(0, 0) = s$, of degree d in both variables, and sends to each party $p_i \in \mathcal{P}$ the (univariate) polynomials $k_i(y) = g(\alpha_i, y)$ and $h_i(x) = g(x, \alpha_i)$.
2. For each pair of parties (p_i, p_j): p_i sends $k_i(\alpha_j)$ to party p_j, and p_j checks whether $k_i(\alpha_j) = h_j(\alpha_i)$. If this check fails, it broadcasts a complaint.
3. For all $k_i(\alpha_j)$, for which no inconsistency was reported, IC-SIGN is invoked once with signer p_j and intermediary p_i to compute the signature $\langle k_i(\alpha_j) \rangle_{j,i}$, and once with signer p_i and intermediary p_j to compute the signature $\langle k_i(\alpha_j) \rangle_{i,j}$.
4. The dealer broadcasts each value for which either an inconsistency was reported (Step 2), or the output of IC-SIGN was $\perp$ (Step 3), and a default signature is used.
5. If some party p_i observes an inconsistency between the polynomials received in Step 1 and the broadcasted values in Step 4, it accuses the dealer. The dealer answers the accusation by broadcasting both $k_i(y)$ and $h_i(x)$. Now, if some other party p_j observes an inconsistency between the polynomial received in Step 1 and these broadcasted polynomials, it also accuses the dealer. This step is repeated until no additional party accuses the dealer. For all broadcasted values, default signatures are used.
6. If the dealer does not answer some complaint or accusation, or if the broadcasted values contradict each other, the parties output a default d-sharing of a default value (with default signatures).
   Otherwise, each party p_i outputs the share $s_i := k_i(0)$, the share polynomial $\hat{s}_i(y) := k_i(y)$ with signatures $\langle \hat{s}_i(\alpha_j) \rangle_{j,i}$ (for $j = 1, \ldots, n$), and the share shares $s_{ji} := h_i(\alpha_j)$ with signatures $\langle s_{ji} \rangle_{j,i}$ (for $j = 1, \ldots, n$). The dealer outputs $\hat{s}(x) := g(x, 0)$.

Fig. 10. The share protocol for threshold adversaries.

Lemma 14. *Let $d < n$ be the sharing parameter. On input s from the dealer,* SHARE *correctly, secretly, and robustly computes a d-sharing. If d is $|\mathcal{D}^*|$-permissive, and if the dealer is correct, the sharing uniquely defines the secret s.*

Proof. SECRECY: It follows from the properties of a bivariate polynomial that $g(x, y)$ reveals no more information about s than the specified output. After Step 1, the adversary does not obtain any additional information: In Step 4, a value s_{ij} is broadcasted only if p_i, p_j or the dealer is actively corrupted, i.e., the adversary knew the value already beforehand. Hence, the protocol does not leak more information than the specified output, and thus always provides secrecy.
CORRECTNESS: First, we have to show that the protocol outputs a valid d-sharing. Due to the bilateral consistency checks, any inconsistency in the values held by correct parties is detected in Step 2 and resolved in Step 4. Therefore, the values held by correct parties uniquely define a polynomial $g'(x, y)$ of degree d, which implies that $g'(x, 0)$ is of degree d. Furthermore, it follows from the properties of IC-SIGN that in Step 3, either a correct IC-signature is computed, or all parties output $\perp$. In the latter case, a default (and hence correct) IC-signature is used. Therefore, the output is a valid d-sharing. Second, we have to show that if d is $|\mathcal{D}^*|$-permissive and if the dealer is correct, then the shared value equals the input of the dealer. A correct dealer can always consistently answer all complains and accusations with the correct values. Hence, if d is $|\mathcal{D}^*|$-permissive, the unique value defined by the sharing is the secret s.
ROBUSTNESS: By inspection, the protocol does not abort. □

The public reconstruction protocol (Fig. 11) proceeds sharewise: For each share s_i, first party p_i broadcasts the share s_i together with the sharing polynomial $\hat{s}_i(y)$, and opens the signatures on all share shares $\hat{s}_i(\alpha_j)$. Second, all parties broadcast their share shares s_{ij}, and open the corresponding signatures. If active corruption took place, these two steps might produce conflicts between certain parties. Note that these conflicts do not only depend on the actively, but also on the passively corrupted parties, due to their inability to reliably verify IC-signatures. If these conflicts can be explained with an adversary corrupting $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq E$, then the share is accepted. Otherwise it is ignored. This technique allows also passively-corrupted parties to reliably verify signatures and therefore reconstruct the correct value. Finally, the secret is reconstructed using the accepted shares. Note that PUBLIC RECONSTRUCTION is the only subprotocol that might abort. All other protocols abort only if they use PUBLIC RECONSTRUCTION as a subprotocol and the invocation thereof aborts. Therefore, it is sufficient to discuss agreement on abort only for this protocol.

Protocol PUBLIC RECONSTRUCTION: Given a d-sharing $[s]$ of some value s, reconstruct s to all parties.

1. For each party p_i:
   (a) p_i broadcasts $\hat{s}_i(y)$ and invokes IC-REVEAL on the signatures $\langle \hat{s}_i(\alpha_j) \rangle_{j,i}$ $(j = 1, \ldots, n)$ of all share shares.
   (b) Each p_j broadcasts its share share s_{ij} and invokes IC-REVEAL on the corresponding signature $\langle s_{ij} \rangle_{i,j}$.
   (c) **Voting:** Each p_k checks whether
      i. the polynomial $\hat{s}_i(y)$ broadcasted in Step 1(a) is consistent with its share share, i.e. $s_{ik} = \hat{s}_i(\alpha_k)$,
      ii. the output of all invocations of IC-REVEAL in Step 1(a) was "accept",
      iii. for all s_{ij} broadcasted in Step 1(b) either $s_{ij} = \hat{s}_i(\alpha_j)$ or the output of IC-REVEAL on the corresponding signature $\langle s_{ij} \rangle_{i,j}$ was "reject".

   p_k broadcasts "yes" if all checks succeed, "no" if check i. or ii. fails, and $\perp$ otherwise. Let a and r denote the number of parties broadcasting "yes" and "no", respectively.
   (d) **Decision:** Accept s_i if $\exists (e_a, e_p) \in E : r \leq e_a \ \wedge \ (e_p + d \geq n \ \vee \ a \geq n - e_p)$. Otherwise ignore s_i.
2. **Output:** If at least $d + 1$ shares are accepted, interpolate these shares with a polynomial $\hat{s}'(x)$ and output $\hat{s}'(0)$. Otherwise abort.

Fig. 11. The public reconstruction protocol for threshold adversaries.

In the voting process, a "yes" means that party p_i (the party currently revealing its share s_i) seems to be correct (which holds unless there are less than $d + 1$ correct parties), and a "no" means that p_i is clearly actively corrupted. A $\perp$ means that the voter does not know which is the case, because there were two or more inconsistent values with valid signatures. Note that a

wrong value with a valid signature may appear in case of an actively corrupted intermediary and either a passively corrupted signer or receiver.

Lemma 15. *Given the sharing parameter d, the reconstruction parameter E, and a d-sharing $[s]$ of some value s,* PUBLIC RECONSTRUCTION *reconstructs s to all parties. The protocol is statistically correct if $|\mathcal{D}^*| < n - d$ and*

$$\forall (e_a, e_p) \in E: \ |\mathcal{D}^*| < n - d - e_a \vee \\ (d + e_p < n \ \wedge \ |\mathcal{D}^*| < n - e_p) \ \vee \ (|\mathcal{E}^*| < n - d \ \wedge \ |\mathcal{E}^*| < n - e_a).$$

Furthermore, it is statistically robust if additionally $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq E$, and always guarantees agreement on abort.

Proof. CORRECTNESS: The protocol outputs a value only if at least $d + 1$ shares are accepted. Trivially, the output is correct if all accepted shares are correct, i.e., when incorrect shares are not accepted. More precisely, we have to show that for any incorrect share $s'_i \neq s_i$ and for each $(e_a, e_p) \in E$, the condition in Step 1(d) is violated. In this proof, we distinguish three cases, depending on which or-term of the condition in the lemma is fulfilled:

i. *Case $|\mathcal{D}^*| < n - d - e_a$:*
In order to broadcast a wrong share $s'_i \neq s_i$, an actively corrupted party p_i has to change the value of at least $n - d$ share shares. At least $n - d - |\mathcal{D}^*|$ of these share shares belong to correct parties that subsequently vote "no", i.e. $r \geq n - d - |\mathcal{D}^*|$. Since $|\mathcal{D}^*| < n - d - e_a$, this implies $r > e_a$, and the share is not accepted.

ii. *Case $d + e_p < n \ \wedge \ |\mathcal{D}^*| < n - e_p$:*
Since $|\mathcal{D}^*| < n - d$, there are at least $d + 1$ correct parties. Hence, in order to broadcast a wrong share $s'_i \neq s_i$, an actively corrupted party p_i has to change the value of at least one share share belonging to a correct party. In Step 1(b), this correct party broadcasts the correct share share with a valid signature, and no correct party accepts the wrong share s'_i, i.e. $a \leq |\mathcal{D}^*|$. Since $|\mathcal{D}^*| < n - e_p$, we have $a < n - e_p$. Since we also have $d + e_p < n$, the share is not accepted.

iii. *Case $|\mathcal{E}^*| < n - d \ \wedge \ |\mathcal{E}^*| < n - e_a$:*
Since $|\mathcal{E}^*| < n - d$, there are at least $d + 1$ honest parties. Hence, in order to broadcast a wrong share $s'_i \neq s_i$, an actively corrupted party has to change the value of at least one share share belonging to an honest party, and to create the signature on this (incorrect) share share. All honest parties notice that this signature is not valid and reject, i.e., $r \geq n - |\mathcal{E}^*|$. Since $|\mathcal{E}^*| < n - e_a$, we have $r > e_a$, and the share is not accepted.

ROBUSTNESS: Given that the correctness condition holds, the protocol guarantees robustness if enough (i.e. $d + 1$) shares are accepted. Let $(e_a, e_p) \in E$ such that $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq (e_a, e_p)$. First, observe that if party p_i is correct, then $r \leq e_a$: All share shares and signatures broadcasted in Step 1(a) are correct and valid. Therefore, no correct party votes "no". Furthermore, if party p_i is honest, then $a \geq n - e_p$: If some p_j broadcasts a contradicting (wrong) share share in Step 1(b), then the signature on this share share is invalid for all honest parties.

It follows from the two observations above that shares from honest parties are always accepted. If $e_p + d < n$, then there are at least $d + 1$ honest parties and the protocol does not abort. Otherwise, if $e_p + d \geq n$, then also shares from correct parties are accepted. Since $|\mathcal{D}^*| < n - d$ there are always at least $d + 1$ correct parties and the protocol does not abort.
AGREEMENT ON ABORT: Since the abort decision is based only on broadcasted values, we always have agreement on abort. □

Addition, Multiplication, and Random Values. Linear functions (and in particular additions) can be computed locally, since d-sharings are linear: Given sharings $[a]$ and $[b]$, and a constant

c, one can easily compute the sharings $[a] + [b]$, $c[a]$, and $[a] + c$. Computing a shared random value can be achieved by letting each party p_i share a random value r_i, and computing $[r] = [r_1] + \ldots + [r_n]$.

For the multiplication of two shared values, we first provide a non-robust multiplication protocol, which we then make robust using *dispute control* [BH06] and *circuit randomization* [Bea91].

Non-robust Multiplication. The product c of two d-shared values a and b is computed as follows [GRR98]: Each party multiplies its shares a_i and b_i, obtaining $v_i = a_i b_i$. This results in a sharing of c with a polynomial $\hat{v}(x)$ of degree $2d$. We reduce the degree by having each party d-share its value v_i (resulting in $[v_i]$), and employing Lagrange interpolation to distributedly compute $[\hat{v}(0)]$, which is a d-sharing of the product c. In order to prevent an active party from sharing a wrong value $v_i' \neq v_i$, each party has to prove in zero-knowledge that $v_i = a_i b_i$. If this proof is not accepted, the non-robust multiplication protocol is aborted.

Protocol MULTIPLICATION$^{\text{NR}}$: Given $[a]$ and $[b]$, compute $[c]$ for $c = ab$.

1. For each party p_i:
   (a) p_i computes $v_i = a_i b_i$, and invokes SHARE on v_i, resulting in $[v_i]$.
   (b) Invoke ABC-PROOF$^{\text{NR}}$ on $\hat{a}_i(x)$, $\hat{b}_i(x)$, and $\hat{v}_i(x)$ (where $\hat{a}_i(x)$ and $\hat{b}_i(x)$ denote the share polynomials of a_i and b_i, respectively, and $\hat{v}_i(x)$ denotes the main polynomial of the sharing $[v_i]$). If the proof is not accepted, the protocol is aborted.
2. All parties distributively compute the Lagrange interpolation on $[v_1], \ldots, [v_n]$ for $c = v(0)$, and output the resulting $[c]$, i.e., $[c] = \sum \lambda_i [v_i]$ for Lagrange coefficients λ_i.

Fig. 12. The non-robust multiplication protocol for threshold adversaries.

Lemma 16. *Given are the sharing parameter d, and d-sharings of a and b. If $2d < n$ and the subprotocols are correct against $(|\mathcal{D}^*|, |\mathcal{E}^*|)$,* MULTIPLICATION$^{\text{NR}}$ *either outputs a correct d-sharing of the product $c = ab$, or it aborts. It aborts only if some party deviates. Furthermore, it is secret against $(|\mathcal{D}^*|, |\mathcal{E}^*|)$ whenever the subprotocols are secret against $(|\mathcal{D}^*|, |\mathcal{E}^*|)$.*

Proof. In Step 2, the parties interpolate a polynomial of degree $2d$ using n evaluation points. Since $2d < n$, this interpolation computes the correct result. Given this observation, it follows by inspection that the protocol is as secure as the subprotocols. □

The ABC-PROOF$^{\text{NR}}$ subprotocol (Fig. 13) allows a prover p to prove that for three shared values a, b, and c it holds that $c = ab$. For this subprotocol, it is sufficient that the values are shared with a simple Shamir sharing, i.e., there is a polynomial $\hat{a}(x)$ with $\hat{a}(0) = a$ that is known to p, and each party $p_j \in \mathcal{P}$ holds $\hat{a}(\alpha_j)$ (for b and c analogously). The protocol is along the lines of the protocol in [CDD+99].

Lemma 17. *Given are the sharing parameter d, and shared polynomials $\hat{a}(x)$, $\hat{b}(x)$, and $\hat{c}(x)$ of degree d that are known to party p. If $\hat{c}(0) = \hat{a}(0)\hat{b}(0)$ and no party deviates, the protocol outputs "accept" (completeness). If $|\mathcal{D}^*| < n - d$ and $\hat{c}(0) \neq \hat{a}(0)\hat{b}(0)$, then, with overwhelming probability,* ABC-PROOF$^{\text{NR}}$ *does not output "accept" (correctness). Furthermore, the protocol is secret against $(|\mathcal{D}^*|, |\mathcal{E}^*|)$ whenever the subprotocols are secret against $(|\mathcal{D}^*|, |\mathcal{E}^*|)$.*

Proof. COMPLETENESS: It follows from inspection and simple arithmetic that if $\hat{c}(0) = \hat{a}(0)\hat{b}(0)$ and no party deviates from the protocol description, the protocol outputs "accept".
CORRECTNESS: If any (correct) party detects an inconsistency and complains (Step 1.c.ii), the protocol outputs "fail". Otherwise, since $|\mathcal{D}^*| < n - d$ (i.e. there are at least $d + 1$ correct parties), both $\hat{b}''(x)$ and $\hat{z}(x)$ are correctly computed. In that case, it follows along the lines of

Protocol ABC-PROOF$^{\mathrm{NR}}$: Given polynomials $\hat{a}(x)$, $\hat{b}(x)$, and $\hat{c}(x)$ of degree d that are known to party p, and where each party $p_j \in \mathcal{P}$ holds $\hat{a}(\alpha_j)$, $\hat{b}(\alpha_j)$, and $\hat{c}(\alpha_j)$, check whether $\hat{c}(0) = \hat{a}(0)\hat{b}(0)$.

1. For each party $p_i \in \mathcal{P}$, carry out a sub-proof:
   (a) p chooses a uniformly random b', computes $c' = ab'$, and invokes SHARE on both b' and c', resulting in sharings $[b']$ and $[c']$ with main polynomials $\hat{b}'(x)$ and $\hat{c}'(x)$, respectively.
   (b) p_i broadcasts a uniformly random challenge r.
   (c) Assisted reconstruction:
      i. p computes and broadcasts the polynomials $\hat{b}''(x) = r\hat{b}(x) + \hat{b}'(x)$ and $\hat{z}(x) = b''\hat{a}(x) - r\hat{c}(x) - \hat{c}'(x)$, where $b'' = \hat{b}''(0)$.
      ii. Each p_j broadcasts a complaint bit indicating whether $\hat{b}''(\alpha_j) \neq r\hat{b}(\alpha_j) + \hat{b}'(\alpha_j)$ or $\hat{z}(\alpha_j) \neq b''\hat{a}(\alpha_j) - r\hat{c}(\alpha_j) - \hat{c}'(\alpha_j)$. If any party complains, output "fail" (and stop the execution).
   (d) Verification: If $\hat{z}(0) = 0$ then the sub-proof is accepted. Otherwise it is rejected.
2. If any of the sub-proofs was rejected, output "reject". Otherwise output "accept".

Fig. 13. A protocol for proving that $c = ab$ for threshold adversaries.

the proof in the case for general adversaries that if $\hat{c}(0) \neq \hat{a}(0)\hat{b}(0)$, then the proof is rejected with overwhelming probability.
SECRECY: The only values revealed during the protocol are the polynomials $\hat{b}''(x)$ and $\hat{z}(x)$. If p is correct, then $\hat{b}''(x)$ is perfectly blinded by $\hat{b}'(x)$, and $\hat{z}(0) = 0$. □

Robust Multiplication. We make the above protocol robust in two steps (Fig. 14): First, using *dispute control* [BH06], we repeatedly invoke MULTIPLICATION$^{\mathrm{NR}}$ on two random values x and y until the subprotocol succeeds. Dispute control is based on the fact that complete protocols abort only if some party deviates from the protocol description, which leads to a detectable dispute with other parties. By keeping track of these disputes, the protocol can be adjusted to limit the number of repetitions. Second, we use *circuit randomization* [Bea91] to compute $[c] = [ab] = (a-x)(b-y) + (a-x)[y] + (b-y)[x] + [xy]$. Given shared values x, y, and z where $z = xy$, this is a linear computation with two public reconstructions of $(a-x)$ and $(b-y)$.

Protocol MULTIPLICATION: Given $[a]$ and $[b]$, compute $[c]$ for $c = ab$.

1. Create sharings $[x]$ and $[y]$ of random values x and y.
2. Invoke MULTIPLICATION$^{\mathrm{NR}}$ on $[x]$ and $[y]$ to compute $[z]$ for $z = xy$.
3. If MULTIPLICATION$^{\mathrm{NR}}$ succeeded, then invoke PUBLIC RECONSTRUCTION on $[a-x]$ and $[b-y]$, and compute and output $[c] = (a-x)(b-y) + (a-x)[y] + (b-y)[x] + [z]$.
4. If MULTIPLICATION$^{\mathrm{NR}}$ did not succeed, then
   (a) Each party $p_i \in \mathcal{P}$ broadcasts its randomness and all messages it has received during the creation of $[x]$ and $[y]$, and MULTIPLICATION$^{\mathrm{NR}}$. Then, each party retraces the execution of these steps locally to detect disputing parties.[12]
   (b) Repeat the protocol, where (1) private channels between any two disputing parties are replaced with broadcast channels, and (2) parties that are in dispute with all other parties are simulated locally on default randomness, and messages towards these parties are broadcasted.

Fig. 14. The robust multiplication protocol for threshold adversaries.

Lemma 18. *Given the sharing parameter d, the reconstruction parameter E, and d-sharings of a and b,* MULTIPLICATION *computes a d-sharing of the product $c = ab$. The protocol guarantees correctness and/or secrecy against $(|\mathcal{D}^*|, |\mathcal{E}^*|)$ whenever the subprotocols provide the corresponding security guarantee against $(|\mathcal{D}^*|, |\mathcal{E}^*|)$. Furthermore, it is robust against $(|\mathcal{D}^*|, |\mathcal{E}^*|)$ whenever* PUBLIC RECONSTRUCTION *is robust against $(|\mathcal{D}^*|, |\mathcal{E}^*|)$.*

[12] Two parties are in dispute if one party claims to have received a message from the other party that is incorrect according to the preceding protocol execution.

Proof. The protocol is repeated until MULTIPLICATION$^{\text{NR}}$ succeeds. In the repetitions where MULTIPLICATION$^{\text{NR}}$ does not succeed, correctness and secrecy of $[x]$, $[y]$, and $[z]$ do not need to be maintained. In the repetition where MULTIPLICATION$^{\text{NR}}$ succeeds, we have $z = xy$ and therefore also $c = ab$. Furthermore, when a private channel is replaced with a broadcast channel because of a dispute, at least one of the two corresponding parties was actively corrupted. Therefore, the secrecy of the subprotocols is maintained. Furthermore, since disputing parties communicate via broadcast channels, each found inconsistency constitutes a new dispute. Hence, the protocol is repeated at most n^2 times. □

The Security of the Parametrized Protocol. Considering the security of the subprotocols described above, we can derive the security of the parametrized protocol, denoted by $\pi^{d,E}$:

Lemma 19. *Let d be the sharing parameter, and E be the reconstruction parameter, the protocol $\pi^{d,E}$ guarantees* statistical correctness *if $d < n - |\mathcal{D}^*|$, $2d < n$, and*

$$\forall (e_a, e_p) \in E: \ |\mathcal{D}^*| < n - d - e_a \vee \\ (d + e_p < n \ \wedge \ |\mathcal{D}^*| < n - e_p) \ \vee \ (|\mathcal{E}^*| < n - d \ \wedge \ |\mathcal{E}^*| < n - e_a).$$

Furthermore, the protocol guarantees statistical secrecy *if additionally $|\mathcal{E}^*| \leq d$, and/or* statistical robustness *if additionally $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq E$.*

Proof. $\pi^{d,E}$ provides a certain security guarantee against $(|\mathcal{D}^*|, |\mathcal{E}^*|)$ if all subprotocols (cf. Lemmas 14 to 18) and the sharing (cf. Lemma 13) provide this guarantee against $(|\mathcal{D}^*|, |\mathcal{E}^*|)$. For each guarantee, it can easily be verified that the condition in the lemma implies the conditions in the corresponding lemmas. □

4.2 Main Result

The following theorem states the optimal bound for statistically secure MPC for threshold adversaries with both mixed adversaries and hybrid security. We show that the bound is sufficient for MPC by providing parameters for the generalized protocols described above. The necessity of the bound follows directly from the corresponding proof for general adversaries (Sec. 3.3).

Theorem 2. *In the secure channels model with broadcast and threshold adversaries, statistically secure (reactive) MPC among $n \geq 2$ parties with multi-thresholds T^c, T^s, T^r, and T^f, where $T^f \leq T^s \leq T^c$ and $T^r \leq T^c$, is possible if*

$$T^s = \{(0,0)\} \quad \vee \\ \forall (t_a^c, t_p^c) \in T^c, (t_a^r, t_p^r) \in T^r, (\cdot, t_p^s), (\cdot, t_p^{s\prime}) \in T^s: \\ t_p^s + t_p^{s\prime} < n \quad \wedge \quad t_p^s + t_a^c < n \quad \wedge \\ \big(t_a^c + t_a^r + t_p^s < n \ \vee \ (t_p^s + t_p^r < n \wedge t_a^c + t_p^r < n) \ \vee \ (t_p^s + t_p^c < n \wedge t_a^r + t_p^c < n)\big)$$

This bound is tight: If violated, there are (reactive) functionalities that cannot be securely computed.

Proof (Sufficiency). If $T^s = \{(0,0)\}$, there is no secrecy requirement, and we can directly use the trivial non-secret protocol described in the Appendix of [HLMR11]. Otherwise, we employ the parametrized version $\pi^{d,E}$ of the protocol of [BGW88] described in Sec. 4.1 with $d := \widetilde{t_p^s}$ and $E := T^r \cup T^f$, where $\widetilde{t_p^s} = \max\{t_p^s \mid (\cdot, t_p^s) \in T^s\}$.

We apply Lemma 19 to derive correctness, secrecy and robustness: Given the bound in the theorem, the choice of the parameters d and E, and the fact that $(|\mathcal{D}^*|, |\mathcal{E}^*|)$ is below the corresponding threshold, it is easy to verify that the condition for each property is fulfilled. In particular, note that the correctness condition is also fulfilled for $(e_a, e_p) \in T^f$: Using that

$T^f \leq T^s$, we have $d + e_p \leq 2\widetilde{t_p^s} < n$ and $e_a + e_p \leq t_a^c + d < n$ (where the inequalities follow from the second line of the condition in the theorem with $t_p^s = t_p^{s\prime} = \widetilde{t_p^s}$).

For fairness, note that $T^f \leq E$. Hence, for $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq (t_a^f, t_p^f)$ the protocol is robust, and the adversary cannot abort. □

5 Conclusion

Our results provide insights into the relations between passive corruption and different security requirements. The bounds presented in this work quantify the impact of passively corrupted parties on all security guarantees. We have shown that, in the statistical setting, passively corrupted parties play a significant role for all security guarantees, and not only for secrecy. Consider the following example: Let $n = 4$, $t_a^c = 2$, $t_p^c = 2$, $t_a^r = 1$, $t_p^r = 2$, and $t_p^s = 1$. For this choice of thresholds, the construction in this paper provides a protocol that is correct and robust (given that the adversary remains below the corresponding thresholds). Yet, we show that it is impossible to construct a protocol that tolerates a single additional passive corruption.

Furthermore, in addition to the known tradeoff between different security guarantees like robustness and correctness [HLMR11], we obtain a novel tradeoff between active and passive corruptions even when only considering a single security guarantee.

Solutions for the setting with general adversaries encompass all possible adversary structures. Yet, these protocols are usually superpolynomial in the number of parties. Therefore, protocols for the setting with threshold adversaries are of more practical relevance. In this work, we provide the first protocol allowing for multi-thresholds, a setting that is strictly more flexible than single-thresholds. This constitutes a substantial step towards general adversaries without losing efficiency in the number of parties.

References

[Bea89] Donald Beaver. Multiparty protocols tolerating half faulty processors. In *CRYPTO '89*, pages 560–572. Springer, 1989.

[Bea91] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO '91*, pages 420–432. Springer, 1991.

[BFH+08] Zuzana Beerliova-Trubiniova, Matthias Fitzi, Martin Hirt, Ueli Maurer, and Vassilis Zikas. MPC vs. SFE: Perfect security in a unified corruption model. In *TCC 2008*, pages 231–250. Springer, 2008.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC '88*, pages 1–10. ACM, 1988.

[BH06] Zuzana Beerliova-Trubiniova and Martin Hirt. Efficient multi-party computation with dispute control. In *TCC 2006*, pages 305–328. Springer, 2006.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols. In *STOC '88*, pages 11–19. ACM, 1988.

[CDD+99] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In *EUROCRYPT '99*, pages 311–326. Springer, 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 *CRYPTO '87*, pages 87–119. Springer, 1988.

[Cha89] David Chaum. The spymasters double-agent problem: Multiparty computations secure unconditionally from minorities and cryptographically from majorities. In *CRYPTO '89*, pages 591–602. Springer, 1989.

[DDWY93] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. *Journal of the ACM*, 40(1):17–47, 1993.

[FHHW03] Matthias Fitzi, Martin Hirt, Thomas Holenstein, and Jürg Wullschleger. Two-threshold broadcast and detectable multi-party computation. In *EUROCRYPT 2003*, pages 51–67. Springer, 2003.

[FHM98] Matthias Fitzi, Martin Hirt, and Ueli Maurer. Trading correctness for privacy in unconditional multi-party computation (extended abstract). In *CRYPTO '98*, pages 121–136. Springer, 1998.

[FHM99] Matthias Fitzi, Martin Hirt, and Ueli Maurer. General adversaries in unconditional multi-party computation. In *ASIACRYPT '99*, pages 232–246. Springer, 1999.

[FHW04] Matthias Fitzi, Thomas Holenstein, and Jürg Wullschleger. Multi-party computation with hybrid security. In *EUROCRYPT 2004*, pages 419–438. Springer, 2004.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC '87*, pages 218–229. ACM, 1987.

[GRR98] Rosario Gennaro, Michael O. Rabin, and Tal Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *PODC '98*, pages 101–111. ACM, 1998.

[HLMR11] Martin Hirt, Christoph Lucas, Ueli Maurer, and Dominik Raub. Graceful degradation in multi-party computation. In *ICITS 2011*, pages 163–180. Springer, 2011.

[HLMR12] Martin Hirt, Christoph Lucas, Ueli Maurer, and Dominik Raub. Passive corruption in statistical multi-party computation (extended abstract). In *ICITS 2012*. Springer, 2012.

[HM97] Martin Hirt and Ueli Maurer. Complete characterization of adversaries tolerable in secure multi-party computation. In *PODC '97*, pages 25–34. ACM, 1997.

[HMZ08] Martin Hirt, Ueli Maurer, and Vassilis Zikas. MPC vs. SFE: Unconditional and computational security. In *ASIACRYPT 2008*, pages 1–18. Springer, 2008.

[IKLP06] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In *CRYPTO 2006*, pages 483–500. Springer, 2006.

[Kat07] Jonathan Katz. On achieving the "best of both worlds" in secure multiparty computation. In *STOC '07*, pages 11–20. ACM, 2007.

[Kil00] Joe Kilian. More general completeness theorems for secure two-party computation. In *STOC '00*, pages 316–324. ACM, 2000.

[LRM10] Christoph Lucas, Dominik Raub, and Ueli Maurer. Hybrid-secure MPC: Trading information-theoretic robustness for computational privacy. In *PODC '10*, pages 219–228. ACM, 2010.

[Mau02] Ueli Maurer. Secure multi-party computation made simple. In *SCN '02*, pages 14–28. Springer, 2002.

[PW92] Birgit Pfitzmann and Michael Waidner. Unconditional byzantine agreement for any number of faulty processors. In *STACS '92*, pages 339–350. Springer, 1992.

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *STOC '89*, pages 73–85. ACM, 1989.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Yao82] Andrew C. Yao. Protocols for secure computations (extended abstract). In *FOCS '82*, pages 160–164. IEEE, 1982.