Diss. ETH No. 20500

Graceful Degradation in Multi-Party Computation

A dissertation submitted to

ETH ZURICH

for the degree of
Doctor of Sciences

presented by

Christoph Lucas
MSc ETH CS, ETH Zurich

born May 03, 1983
citizen of the Federal Republic of Germany

accepted on the recommendation of

Prof. Dr. Ueli Maurer, examiner
Prof. Dr. Jesper Buus Nielsen, co-examiner
Dr. Martin Hirt, co-examiner

2012

Acknowledgments

First of all I would like to thank Ueli, most importantly for giving me the great opportunity to do a PhD in his group and to explore the world of research. Since my first year at ETH, when I had the pleasure to attend one of his lectures, I have been impressed by his way of teaching and thinking. During my master studies, he introduced me to the realm of cryptography, which has been a passion of mine ever since. Thanks to him I understood the power of abstraction as a means of simplification, not only in research, but also in real life.

Secondly, I am very thankful to Martin. Without his patience in our many discussions, I would certainly not have reached this point. From him I learned the tedious but fascinating process of turning an intuitive idea into science. I have to admit that in our often controversial research discussions he was mostly right. Yet, he was always kind enough to let me win when playing pool afterwards. I hope there are still plenty of pool games ahead of us.

Furthermore, I thank Jesper for serving on my committee, for reviewing my thesis, and for his valuable feedback. Special thanks go also to my coauthor Nik, with whom I had many interesting discussions especially at the beginning of my PhD, and to our secretary Beate for her administrative support.

I would like to express my deepest gratitude to my parents, Gert and Sabine. From my very childhood on, they taught me to be curious, to ask questions, and to be open when looking for answers. I hope I picked up at least some of their teachings. Throughout my time at school and later at university, they gave me all the support a kid could dream of, even during times when it was not easy for them. I know that wherever I am, whatever I do, I can count on them. This gives me the confidence to set out on enterprises that seem impossible in the first place.

Last but by no means least, I would like to thank my friends who were there for me when times got rough and who got my mind on to other things. In particular, there were two persons that gave me their exceptional personal support. I might have given up if not for their understanding and encouragement: Sophie, thank you for being in my life and for letting me be part of your life. Björn, thanks for your countless helpful advices, being it research related or not.

The research in this thesis was carried out under partial support from the Zurich Information Security Center (ZISC).

"No sólo no hubiéramos sido nada sin ustedes, sino con toda la gente que estuvo a nuestro alrededor desde el comienzo; algunos siguen hasta hoy. ¡Gracias totales!"

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. Security can be maintained against more passive corruptions than is possible for active corruptions.

Most MPC protocols provide security guarantees in an *all-or-nothing* fashion: Either the set of corrupted parties is tolerated and the protocol provides all specified security guarantees, or the set of corrupted parties is not tolerated and the protocol provides no security guarantees at all. Similarly, corruptions are in an all-or-nothing fashion: Either a party is fully honest, or it is fully corrupted. For example, an actively secure protocol is rendered completely insecure when just one party is corrupted additionally to what is tolerated, even if all corrupted parties are only passive.

In this thesis, we provide the first treatment of MPC with graceful degradation of both security and corruptions. First of all, our protocols provide graceful degradation of security, i.e., different security guarantees depending on the actual number of corrupted parties: the more corruptions, the weaker the security guarantee (so-called *hybrid* security). We consider all security properties generally discussed in the literature (secrecy, correctness, robustness, fairness, and agreement on abort). Furthermore, the protocols provide graceful degradation with respect to the corruption type, by distinguishing fully honest parties, passively corrupted parties, and actively corrupted parties (so-called *mixed* adversaries).

We prove exact bounds for which MPC with graceful degradation of security and corruptions is possible for both threshold and general adversaries and for all security levels (perfect, statistical, and computational). Furthermore, we provide protocols that meet these bounds. This strictly generalizes known results on hybrid security and mixed adversaries.

Among our technical contributions, especially two might be of independent interest: First, we introduce the notion of *multi-thresholds*. To the best of our knowledge, all known protocols for threshold mixed adversaries 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. Second, we present a new secret-sharing scheme that, in the reconstruction phase, releases secrecy *gradually*. This allows to construct non-robust MPC protocols which, in case of an abort, still provide as much secrecy as possible.

Zusammenfassung

Das Ziel von Mehr-Parteien-Berechnungen ist es, eine beliebige Berechnung verteilt, geheim und fehlertolerant durchzuführen. Dafür führt eine gegebene Menge von n Parteien ein Protokoll aus, das gewisse Sicherheitsgarantien wie zum Beispiel Korrektheit, Geheimhaltung, Robustheit und Fairness erhält, auch wenn eine Teilmenge der Parteien von einem Gegner korrumpiert ist. Korruptionen sind entweder *passiv* oder *aktiv*: Eine passiv korrumpierte Partei führt das Protokoll wie eine ehrliche Partei korrekt aus, der Gegner erfährt allerdings den kompletten internen Zustand dieser Partei. Eine aktiv korrumpierte Partei wird vollständig vom Gegner kontrolliert und kann beliebig vom Protokoll abweichen. Protokolle können Sicherheit gegen mehr passiv als aktiv korrumpierte Parteien garantieren.

Die meisten Protokolle für Mehr-Parteien-Berechnungen geben entweder alle oder keine Sicherheitsgarantien: Entweder wird die Menge der korrumpierten Parteien toleriert und das Protokoll garantiert alle spezifizierten Sicherheitseigenschaften, oder die Menge der korrumpierten Parteien wird nicht toleriert und das Protokoll gibt keine Sicherheitsgarantien. Das gleiche gilt für Korruptionen: Eine Partei ist entweder vollkommen ehrlich oder vollkommen korrumpiert. Zum Beispiel ist ein Protokoll, das sicher ist gegen aktive Korruptionen, komplett unsicher wenn auch nur eine einzige Partei mehr korrumpiert ist als toleriert wird, selbst wenn alle korrumpierten Parteien nur passiv korrumpiert sind.

In dieser Doktorarbeit behandeln wir zum ersten Mal Mehr-Parteien-Berechnungen mit „Graceful Degradation“ (z. Dt. fortschreitende Verschlechterung) sowohl der Sicherheit als auch der Korruptionen. Erstens bieten unsere Protokolle „Graceful Degradation“ der Sicherheit, d. h. die Sicherheitsgarantien nehmen mit steigender Anzahl korrumpierter Parteien graduell ab: je mehr Parteien korrumpiert sind, desto schwächer sind die Sicherheitsgarantien (sogenannte *hybride* Sicherheit). Wir betrachten alle Sicherheitseigen-

schaften, die gewöhnlich in der Literatur diskutiert werden (Geheimhaltung, Korrektheit, Robustheit, Fairness und Einigkeit über den Protokollabruch). Zweitens bieten die Protokolle „Graceful Degradation" der Korruptionen, d. h. die Sicherheitsgarantien hängen nicht von der Gesamtanzahl der Korruptionen ab, sondern davon, wieviele Parteien mit welchem Typ korrumpiert sind. Wir betrachten vollkommen ehrliche Parteien, passiv korrumpierte Parteien und aktiv korrumpierte Parteien in einem einzigen Protokolllauf (sogenannte *gemischte* Gegner).

Wir beweisen exakte Schranken, wann Mehr-Parteien-Berechnungen mit „Graceful Degradation" möglich sind, sowohl für Schwellwertgegner (also Gegner, die durch einen Schwellwert an die Anzahl korrumpierter Parteien charakterisiert sind) als auch für allgemeine Gegner (also Gegner, die durch eine allgemeine Bechreibung charakterisiert sind) und für alle Sicherheitsniveaus (perfekt, statistisch und berechenmässig). Ausserdem beschreiben wir Protokolle, die diese Schranken erreichen. Diese Resultate sind strikte Verallgemeinerungen von bekannten Resultaten bezüglich hybrider Sicherheit und gemischter Gegner.

Insbesondere zwei der in dieser Arbeit beschriebenen technischen Beiträge könnten von unabhängigem Interesse sein: Erstens führen wir die Idee der Mehrfachschwellwerte ein. So weit uns bekannt ist, charakterisieren alle bekannten Protokolle für gemischte Schwellwertgegner die tolerierten Gegner mit einem einzigen Paar von Schwellwerten (ein Schwellwert für die Anzahl aktiv und ein Schwellwert für die Anzahl passiv korrumpierter Parteien). Dieses Paar beschreibt den einzigen maximalen Gegner, der toleriert wird. Wir verallgemeinern diese einfache Charakterisierung, um mehrere unvergleichbare maximale Gegner berücksichtigen zu können. Tatsächlich können wir im Modell mit Mehrfachschwellwerte Protokolle konstruieren, die strikt mehr Gegner tolerieren, als es mit einem einzigen Paar von Schwellwerten möglich gewesen wäre, und die trotzdem effizient sind. Zweitens präsentieren wir ein neues Verfahren zum Verteilen geheimer Werte, welches in der Rekonstruktionsphase die Geheimhaltung schrittweise abbaut. Mit diesem Verfahren ist es möglich, Protokolle für Mehr-Parteien-Berechnungen zu konstruieren, die zwar nicht robust sind, aber bei einem Abbruch die bestmögliche Geheimhaltung garantieren.

Contents

Chapter 1

Overview

Imagine you travel to some new place, and you meet some really nice people. Then, during dinner, the radio plays "I've been looking for freedom" from David Hasselhoff, your favorite song when you were a teenager. Of course, you are far too embarrassed to admit that you have ever listened to this song, let alone that David Hasselhoff was your idol. But then again, if the others were also fans, it would be nice to talk about the good old times. Maybe, you could even start some kind of Hasselhoff-You Tube-Party and have the revival night of the year. Still, the embarrassment prevails.

One solution could be to ask the help of the waiter: He could go around and ask each one of the group whether he was a David Hasselhoff fan or not. Then, if everybody admitted his youthful folly, the waiter could let everybody know (or simply play the entire album). In contrast, if one or more of the people at the table said they were not Hasselhoff fans, the waiter should simply shake his head and bring the dessert.

Yet, the waiter might just laugh at each and single one who was a fan. Or he could be the friend of one person in the group, and tell him afterwards who suffered a lapse of taste when he was young. Short, the waiter might not be trusted, or at least not by everybody. So, the question remains how the "David Hasselhoff Problem" can be solved.

If only a (trusted version of the) waiter could be emulated among the people at the table.

1.1 Multi-Party Computation

Considering the above example more formally, there are 5 to 6 parties (the people at the table), each with one bit of input (whether he or she was a Hasselhoff-fan or not). However, the parties do not trust each other, and each party does not want the others to know his or her input. The goal is to compute the logical 'and' of all inputs. This problem can be solved with a protocol for *Multi-Party Computation* (MPC).

More generally, MPC allows a set of n parties to securely compute any (probabilistic) function f in a distributed manner. Security means that secrecy of the inputs and correctness of the output are maintained even when some of the parties are dishonest. Furthermore, such dishonest parties should not be able to abort the protocol once it has started (robustness). If robustness cannot be guaranteed, at least the protocol should be fair in the sense that if the dishonest parties learn the output then so do the remaining parties.

The dishonesty of parties is modeled with a central adversary who corrupts parties. This models the strongest possible adversary, namely one that coordinates the attack of all corrupted parties. The adversary can corrupt parties *passively*, i.e., can read their internal state, or *actively*, i.e., can additionally make them deviate arbitrarily from the protocol. The distinction between different corruption types reflects the real world fact that an attacker might be able to compromise machines in different ways, and that the adversary has to invest more resources to completely control a machine (active corruption), than to just read out the internal memory (passive corruption).

MPC was originally proposed by Yao [Yao82]. The first general solution was given in [GMW87], where two protocols are presented, one providing passive security against any number of corruptions (i.e., $t < n$ corrupted parties), and one providing active security against a corrupted minority (i.e., $t < n/2$). These protocols are computationally secure only, that means based on the computational assumption that some mathematical problem cannot be solved efficiently.

In [BGW88, CCD88], information-theoretic security (i.e., security not based on a computational assumption) was achieved 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 early results considered only *threshold adversaries*. That means, the protocols are characterized by a threshold t which represents the maximum

tolerated number of corruptions against which security can still be guaranteed. The idea behind this symmetric characterization is that each party is equally likely to be corrupted, and that essentially only the number of corrupted parties is of interest.

The model with threshold adversaries is very restrictive, allowing to model only a very specific class of adversaries. The most general characterization is to state for each subset of the set of all parties, whether security is still guaranteed if this subset is corrupted. This model is called *general adversaries*. In this model, the tolerated sets of corrupted parties are 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 [HM97]. Security is guaranteed as long as the set of corrupted parties is one of the subsets in $\mathcal{Z}$.

1.2 Graceful Degradation

All protocols mentioned above achieve full security, i.e., secrecy, correctness, and robustness. Furthermore, those protocols (and most MPC protocols in the literature) do not degrade very gracefully. They provide a very high level of security up to some threshold t (usually full security), but no security at all beyond this threshold. There are no intermediate levels of security. Furthermore, a party is considered either fully honest or fully corrupted. There are no intermediate levels of corruptions.

Many papers in the literature consider several corruption types, or even several levels of security, but in separate protocols. For example, [BGW88] proposes a protocol for passive security with $t < \frac{n}{2}$, and another protocol for active security with $t < \frac{n}{3}$. There is no graceful degradation: If in the active protocol, some *passive* adversary corrupts $\lceil \frac{n}{3} \rceil$ parties, the protocol is insecure.

The notion of graceful degradation was first considered by Chaum [Cha89]. On the one hand, he introduced graceful degradation of security (often called *hybrid* security). Previously known protocols provided either full security, or no security guarantees at all. The idea of hybrid security is to guarantee each security property independently against as many corrupted parties as possible. Chaum proposed a protocol that provides information-theoretic security as long as only a few parties are corrupted, computational security if more parties are corrupted, and no security in case of many corruptions. This protocol was recently generalized in [FHHW03, FHW04, IKLP06, Kat07, LRM10].

On the other hand, Chaum considered graceful degradation of corruptions (often called *mixed* adversaries). Traditional results consider only a single corruption type. Chaum proposed a protocol that considers fully honest, passively corrupted, and actively corrupted parties in the same protocol execution. This protocol was generalized and extended in [DDWY93, FHM98, FHM99, BFH+08, HMZ08].

1.3 Contributions

In this thesis, we introduce the notion of graceful degradation in several dimensions, i.e., we present protocols with graceful degradation of both security (i.e., hybrid security) and of corruptions (i.e., mixed adversaries). Hybrid security and mixed adversaries are two concepts with the same goal: To provide protocols with fine-grained security guarantees and high flexibility that model real life requirements as closely as possible. While they follow two orthogonal approaches on a conceptual level, similarities can be found on a technical level. Consequently, it is only natural to merge these two directions of research and to provide a single model with both types of graceful degradation.

We solve the problem of graceful degradation of security and of corruptions in the perfect, the statistical, and the computational setting, for both threshold and general adversaries. Our main results are strict generalizations of the previous results for MPC, which appear as special cases in our unified treatment.

Apart from the main goal to provide these protocols (together with tight bounds), we developed several techniques which might be of independent interest. Furthermore, our results give precise insights that were previously only conjectured on an intuitive level.

Multi-thresholds: The model with general adversaries is the most general possible model. Yet, the runtime of protocols is polynomial in the size of the adversary structure, which is (typically) exponential in the number of parties. In contrast, the runtime of protocols for threshold adversaries is (usually) polynomial in the number of parties. However, the threshold model imposes a strong restriction on the ability to characterize the tolerated adversaries. In this thesis, we introduce the model with multi-thresholds (Section 3.3). The runtime of protocols in this model is also polynomial in the number of parties, yet multi-thresholds can capture strictly more adversaries than simple thresholds: Consider two adversaries, one that corrupts a few parties actively and

many parties passively, and another that corrupts more parties actively, but less parties passively. These two adversaries are incomparable, i.e., security against one of the two does not imply security against the other. Now, assume that there is a protocol that tolerates both adversaries. Simple thresholds do not allow to make a security statement that encompasses both tolerated adversaries, while multi-thresholds do.

Gradual VSS: Most protocols for verifiable secret sharing (VSS) from the literature show a very unfair behaviour: Even if the adversary aborts the reconstruction protocol, he usually still learns the shared secret. Fairness means that, in every step, the adversary either did not yet obtain any information about the secret (secrecy), or cannot abort the protocol (robustness). We introduce the notion of gradual VSS, which reduces secrecy in a step-wise fashion and at the same time increases robustness (Section 6.1). This results in VSS schemes that provide the best possible fairness.

In most traditional MPC protocols with mixed adversaries, there is a static tradeoff between active and passive corruptions: the higher the threshold for active corruptions, the lower the threshold for passive corruptions. That means, only certain pairs of thresholds for active and passive corruptions can be tolerated, and one has to fix the pair before the protocol execution. The combination of a gradual VSS with multi-thresholds allows to model a *dynamic* tradeoff, where these thresholds do not have to be fixed beforehand (Section 6.3). As a consequence, the protocol is secure against all tolerated pairs of thresholds for active and passive corruptions simultaneously.

Group Commitments: For the setting with statistical security, we introduce the notion of *group commitments*, which is a pair of protocols that allows a group to first commit to a value on which they agree (while providing secrecy with respect to the remaining parties), and, at a later point in time, to reveal this value to the remaining parties (Section 5.2.1.1). While standard commitments are impossible to be realized in this setting, the combination of several parties into a group results in a very useful variant of commitments. From a more technical point of view, this generalization to groups generates more information on which to base the decision in the reveal protocol. The implementation illustrates how such a broader and more complex basis for decision-making can be exploited. We use group commitments instead of IC signatures to construct a verifiable secret sharing scheme, that can be verified reliably even without an honest majority.

Relation between corruption types and security properties: Usually, the intuition behind the different corruption types is that passively corrupted parties only aim to break secrecy, whereas actively corrupted parties aim to

break correctness (and/or robustness). However, this analogy does not readily extend to mixed adversaries that simultaneously perform passive and active corruptions. Our model of graceful degradation separates the different security properties, and therefore allows, for the first time, to make precise statements formalizing the above intuition. While we could confirm this folklore belief for the perfect setting, it turns out that in the statistical setting, passively corrupted parties play an important role not only for secrecy but also for correctness and robustness. Furthermore, in the computational setting, passively corrupted parties influence also fairness (but not correctness or robustness).

Chapter 2

Introduction

In this chapter we discuss the basics of Multi-Party Computation, and make the concepts introduced in Chapter 1 more precise. As a first step, we describe the basic setting (Section 2.1). Then, we provide a detailed description of the adversary model (Section 2.2). Finally, we formalize the notion of security (Section 2.3).

2.1 Parties and Channels

We consider a set of n parties $\mathcal{P} = \{p_1, \ldots, p_n\}$, who want to compute a function f over a finite field $\mathbb{F}$. In general, this function can be probabilistic (i.e., use randomness during its computation), and reactive (i.e., parties can provide further inputs after having received some intermediate outputs). We represent f as a circuit with input, addition, multiplication, random, and output gates. Output gates can be either public (i.e., the value is given to all parties) or private (i.e., it is given only to a specific party).

In a setting with reactive functions, the circuit is partitioned into phases. In each phase, first all input gates are processed, then all computation gates, and finally all output gates. All input gates (as well as all output gates) of a phase are processed in parallel. Note that non-reactive functions can be considered as reactive functions with a single phase.

In all our protocols, we apply two generic simplifications. First, private output, where a value s is disclosed only to a single party p_k, is reduced

to public output using a simple blinding technique [CDG88]: Party p_k first shares a uniform random value, which is added to s before public output is invoked. It is easy to see that p_k can recover s, while all other parties obtain no information about s. Second, we reduce randomness gates to addition gates as follows: First each party p_i inputs a random value r_i. Then, all r_i's are added up. As long as secrecy is guaranteed, this results in a random value. In a setting without secrecy, we do not require true randomness (see Section 2.3.2). As a consequence, we do not need to provide implementations neither for private output, nor for randomness gates.

For the communication between the parties, we assume a complete network of *secure channels*, i.e., each pair of parties can communicate in a confidential (i.e., only the recipient can read the messages), authenticated (i.e., messages cannot be changed during transmission), and robust (i.e., messages cannot be deleted) way. Note that, in the computational setting, confidentiality and authenticity are usually established via standard techniques such as encryption and digital signatures from insecure channels.

A *broadcast channel* allows one party to send a message to all other parties, and all parties are guaranteed to receive the same message (consistency). If the broadcast channel is authenticated, than the message received by all parties is identical to the one sent by the sender (validity). In most of our results, we assume that each party has an authenticated broadcast channel to all other parties. In the computational setting, broadcast channels are usually implemented with an appropriate protocol [DS82].

Furthermore, we consider a *synchronous* model of communication. That means, the protocol execution is split into rounds, and messages sent during one round are guaranteed to reach the recipient before the beginning of the next round. Hence, if a party does not receive a message it is supposed to receive, it can be sure that the sender did not send it (and hence knows that the sender is actively corrupted). For this reason (and 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. We assume that the adversary is *rushing*. That means, in any given round, the adversary receives the messages sent by other parties in that round before sending his own messages.

2.2 The Adversary

The dishonest behaviour of parties is modeled with a central adversary who corrupts certain parties. In Section 2.2.1, we discuss different possible corruption types. Then, we describe how the sets of corrupted parties that a protocol can tolerate without losing security can be characterized; first for the setting with a single corruption type (Section 2.2.2), then for the setting with several corruption types (also called mixed adversary, Section 2.2.3).

2.2.1 Corruption Types

The central adversary corrupts certain parties in $\mathcal{P}$. The implication of such a corruption depends on the corruption type. In this work, we consider the two most prominent corruption types:

- **Passive corruption:** The adversary can read the entire internal state of passively corrupted parties throughout the protocol execution. He learns both all messages the party receives, as well as all the randomness the party chooses. Yet, passively corrupted parties follow the protocol description correctly. Furthermore, we assume that the randomness is chosen "on the fly", whenever needed ("instant randomness"). This allows passively corrupted parties to e.g. select challenges that are unpredictable to the adversary. The set of actually passively corrupted parties during a protocol run is denoted by $\mathcal{E}^*$ (eavesdropping).

- **Active corruption:** The strongest possible notion of corruption is active corruption. Actively corrupted parties are completely controlled by the adversary. That means, he learns their entire internal state, and can make them deviate from the protocol description. Hence, an actively corrupted party might send wrong messages, or no messages at all. The set of actually actively corrupted parties during a protocol run is denoted by $\mathcal{D}^*$ (disrupting).

Uncorrupted parties are called *honest*, non-actively corrupted parties are called *correct*. As a matter of fact, the sets $\mathcal{D}^*$ and $\mathcal{E}^*$ are not known to the parties. In particular, this implies that passively corrupted parties are not aware of the fact that they are corrupted.

2.2.2 Description of the Tolerated Sets of Corrupted Parties

The security of a protocol depends on the set of actually corrupted parties, i.e., a protocol tolerates only certain sets of corrupted parties without losing security. In this work, we consider the two most common approaches to characterize these tolerated sets.

The most general possible approach is to enumerate the tolerated sets of corrupted parties in a so-called adversary structure $\mathcal{Z}$, a collection of subsets of the player set [HM97]. This model is called *general adversaries*. We require that, if a protocol tolerates some subset M, then it must also tolerate all subsets M' of M.[1] Therefore, the adversary structure must be monotone in the sense that for each set $M \in \mathcal{Z}$, all subsets of M must also be in $\mathcal{Z}$ (i.e., $\forall M, M' \subseteq \mathcal{P} : M \in \mathcal{Z} \ \wedge \ M' \subseteq M \Rightarrow M' \in \mathcal{Z}$). We say a set M is *maximal* in $\mathcal{Z}$ if there is no other set M' in $\mathcal{Z}$ with $M \subset M'$. A protocol is secure with respect to $\mathcal{Z}$, if the protocol is secure against all adversaries corrupting parties in some set $M \in \mathcal{Z}$.

The characterization of protocols with general adversaries allows to state for each subset of $\mathcal{P}$ whether security is guaranteed or not in the case where this subset is corrupted. Therefore, this characterization is the most expressive one possible. While protocols for general adversaries are efficient in the input length (which includes the adversary structure $\mathcal{Z}$), the length of $\mathcal{Z}$ itself is usually super-polynomial in the number n of parties.

In the model with *threshold adversaries*, the tolerated sets are characterized with a threshold t: All sets of size at most t are tolerated. In other words, if a protocol is secure with respect to t, it is secure as long as at most t parties are corrupted; and if more than t parties are corrupted, no security guarantees are given. Protocols for threshold adversaries are (usually) efficient in the number of parties, which is in contrast to protocols in the general model.

2.2.3 Mixed Adversaries

Traditional protocols consider only purely active or purely passive adversaries, i.e., all corruptions of the central adversary are of the same type. In order to analyze active and passive corruptions in a single protocol execution, one has to consider *mixed adversaries* that can perform each corruption with a different type. That means that there is one set $\mathcal{D}^*$ of actively corrupted parties, and at the same time another set $\mathcal{E}^*$ of passively corrupted parties. For

[1] Intuitively, the adversary could simply have the parties in $M \setminus M'$ behave honestly.

ease of notation, we assume that the set of actively corrupted parties $\mathcal{D}^*$ is a subset of the set of passively corrupted parties $\mathcal{E}^*$, i.e. that $\mathcal{D}^* \subseteq \mathcal{E}^*$.

In the setting with general adversaries, a protocol for mixed adversaries is characterized by an adversary structure consisting of tuples $(\mathcal{D}, \mathcal{E})$ of subsets of $\mathcal{P}$ with $\mathcal{D} \subseteq \mathcal{E}$, where the parties in $\mathcal{E}$ can be passively corrupted, and the parties in $\mathcal{D} \subseteq \mathcal{E}$ even actively. Again, we assume that the adversary structure is monotone, i.e., $\forall(\mathcal{D}, \mathcal{E}), (\mathcal{D}', \mathcal{E}') : (\mathcal{D}, \mathcal{E}) \in \mathcal{Z} \wedge D' \subseteq D \wedge E' \subseteq E \Rightarrow (\mathcal{D}', \mathcal{E}') \in \mathcal{Z}$. Then, a protocol is secure with respect to $\mathcal{Z}$, if it is secure against all adversaries corrupting some $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}$.

In the threshold setting, a protocol for mixed adversaries is characterized by two thresholds t_a and t_p, where up to t_p parties can be passively corrupted, and up to t_a of these parties can even be corrupted actively. Since $\mathcal{D}^* \subseteq \mathcal{E}^*$, we have that t_p denotes the upper bound on the total number of corruptions (active as well as purely passive), and t_a denotes the upper bound on the number of actively corrupted parties (hence, $t_a \leq t_p$). A protocol is secure with respect to (t_a, t_p), if it is secure as long as $|\mathcal{D}^*| \leq t_a$ and $|\mathcal{E}^*| \leq t_p$ (for which we use the shorthand $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq (t_a, t_p)$).

2.3 Security

In this section we define what it means for a protocol to be secure. We first provide a description of the various security properties (Section 2.3.1), and introduce the notion of hybrid security (Section 2.3.2). Then, we formalize the notion of security (Section 2.3.4), and discuss different levels of security (Section 2.3.5).

2.3.1 Security Properties

We first give a description of the five standard security guarantees, which a protocol can provide.

- *Secrecy* means that the adversary learns nothing neither about the honest parties' inputs and outputs, nor about intermediate results of the computation (except, of course, for what can be derived from the corrupted parties' inputs and outputs).

- *Correctness* means that each (correct) party either outputs the right value or no value at all (in particular, some parties may output the correct value, while others output no value).

- *Robustness* means that the adversary cannot prevent the correct parties from learning their respective outputs. This requirement turns out to be very demanding. Therefore, relaxations have been proposed, where robustness is replaced by weaker output guarantees:

- *Fairness* means that the adversary can possibly prevent the correct parties from learning their outputs. Yet, if not all correct parties learn their complete output, the adversary obtains no information about any output. In the case of reactive MPC (i.e., MPC with multiple, non-concurrent outputs), fairness can only be achieved for outputs provided in the same phase: either all (correct) parties learn their outputs, or also the adversary does not obtain any information about any output. However, the adversary can abort the protocol after having received outputs from prior phases.

- *Agreement on abort* means that the adversary can possibly prevent correct parties from learning their output, even while corrupted parties do learn their outputs, but then the correct parties at least reach agreement on this fact (and typically make no output).

In all our constructions, all abort decisions are based on publicly known values. Hence, we have agreement on abort for free. Note that the impossibility proofs hold even when agreement on abort is not required.

Clearly, actively corrupted parties do not have any security guarantees, as they are completely controlled by the adversary. In contrast, passively corrupted parties may or may not have security guarantees, depending on the interpretation of passive corruption. Our definitions explicitly include guarantees for passively corrupted parties.

2.3.2 Hybrid Security

Most protocols in the literature consider only full security: Each set of corrupted parties is either tolerated (i.e., the protocol provides full security with correctness, secrecy, and robustness), or not (i.e., if a single party is corrupted in addition to what is tolerated, immediately all security guarantees are lost).

Protocols with hybrid security consider each security property separately, and provide each against as many corrupted parties as possible. In other words, the guaranteed security properties (secrecy, correctness, fairness, robustness, agreement on abort) depend on the set of corrupted parties. The more parties are corrupted, the fewer properties are guaranteed. Hence, the security of a protocol is not characterized with a single parameter, but with one parameter for each security property.

In the setting with general adversaries, hybrid security is modeled with four adversary structures: $\mathcal{Z}^c$ for correctness, $\mathcal{Z}^s$ for secrecy, $\mathcal{Z}^r$ for robustness, and $\mathcal{Z}^f$ for fairness. Since all our protocols achieve agreement on abort for free, we do not introduce a separate structure for this security property. Then, correctness is guaranteed if the set of corrupted parties is contained in $\mathcal{Z}^c$, secrecy if it is contained in $\mathcal{Z}^s$, robustness if it is contained in $\mathcal{Z}^r$, and fairness if it is contained in $\mathcal{Z}^f$. If the set of corrupted parties is contained in several structures, all corresponding security properties are achieved. In particular, full security is achieved if the set of corrupted parties is contained in all structures. We assume that $\mathcal{Z}^s \subseteq \mathcal{Z}^c$ and $\mathcal{Z}^r \subseteq \mathcal{Z}^c$, since secrecy and robustness are not well defined in a setting without correctness. Furthermore, we assume that $\mathcal{Z}^f \subseteq \mathcal{Z}^s$ since in a setting without secrecy the adversary inherently has an unfair advantage over honest parties.

In the threshold setting, we consider protocols with four thresholds: t^c for correctness, t^s for secrecy, t^r for robustness, and t^f for fairness. Again, we do not introduce a separate pair of thresholds for agreement on abort. Then, correctness is guaranteed for up to t^c corrupted parties, secrecy is guaranteed for up to t^s corrupted parties, robustness is guaranteed for up to t^r corrupted parties, and fairness is guaranteed for up to t^f corrupted parties. If the number of corruptions is below multiple thresholds, all corresponding security properties are achieved. Analogously to the setting with general adversaries, we assume that $t^s \leq t^c$ and $t^r \leq t^c$, as well as $t^f \leq t^s$.

Note that the notion of correctness for a security level without secrecy differs from the usual interpretation: The adversary is rushing (cf. Section 2.1) and may know the entire state of the protocol execution. Hence, for inputs provided in the same phase, input-independence is not guaranteed. Furthermore, for the same reason, when computing a probabilistic function in a setting without secrecy, we allow the adversary to choose the randomness.

2.3.3 Ideal Functionalities

The circuit to be computed and the desired security properties are specified using an ideal functionality $\mathcal{F}$ that communicates with the parties and performs computations. Ideal functionalities are either deterministic (i.e., perform computations only on the inputs given by parties) or probabilistic (i.e., additionally choose random values). Furthermore, ideal functionalities are either non-reactive or reactive. A *non-reactive* functionality first receives all inputs, then performs all computations, and then provides all outputs. The setting of non-reactive functionalities is also called *Secure Function Evaluation* (SFE). In contrast, *reactive* functionalities are not restricted to a single phase of inputs and outputs. That means, parties can provide additional inputs after having received (intermediate) results.

Before the execution begins, $\mathcal{F}$ is initialized with the set of corrupted parties (i.e., $\mathcal{F}$ is corruption aware), and the adversary structures (or, in the threshold setting, the thresholds) for each security property. Then, $\mathcal{F}$ performs the computation according to the given circuit. If the set of corrupted parties is not contained in one of the adversary structures, $\mathcal{F}$ provides the adversary with additional powers, depending on which security property is not guaranteed. For this purpose, $\mathcal{F}$ has a special adversarial interface, over which it communicates directly with the adversary. For example, if $(\mathcal{D}^*, \mathcal{E}^*) \notin \mathcal{Z}^s$, i.e., if secrecy is not guaranteed, $\mathcal{F}$ forwards all inputs, outputs, intermediate results, etc. to the adversary.

2.3.4 Definition of Security

In the following, we describe the simulation based security model as introduced to MPC by [GMW87, Can00]. The security of a protocol π (the real world) is defined with respect to an ideal functionality $\mathcal{F}$ that correctly performs all computations (the ideal world). Informally, a protocol is secure if whatever an adversary can achieve in the real world, he could also achieve in the ideal world.

More precisely, in the *real world*, for each correct party p_i there is an interactive program π_i that can communicate via (secure bilateral and broadcast) channels with other programs. Furthermore, there is an adversary $\mathcal{A}$. For actively corrupted parties, $\mathcal{A}$ has direct access to the channels. For passively corrupted parties p_j, π_j remains in place, yet $\mathcal{A}$ learns its entire internal state throughout the protocol execution. At the beginning of the execution, each

program π_i takes an input x_i. Also, the adversary takes some input. We denote the vector of all inputs with $\vec{x}$. The input of actively corrupted parties to the computation is implicitly defined by $\mathcal{A}$ and is therefore not part of $\vec{x}$ (i.e., the corresponding entries in the vector are fixed to $\square$). Then, the protocol is executed. At the end of the execution, each program π_i provides all its outputs, and the adversary outputs some function of its internal state (e.g., the identity function). The vector of the output values of all parties (where the output of actively corrupted parties is fixed to $\square$) together with the output of the adversary is denoted by $\text{REAL}^{\vec{x}}_{\pi,\mathcal{A}}$.

The *ideal world* consists of the ideal functionality $\mathcal{F}$ and an ideal adversary (or simulator) σ connected to $\mathcal{F}$ via the adversarial interface. The task of σ is to make the ideal world look identical to the real world, i.e., it mimics the actions of the real world adversary $\mathcal{A}$ as good as possible (in particular, σ depends on $\mathcal{A}$). At the beginning of the execution, $\mathcal{F}$ takes some input x_i for each correct party p_i, and the simulator σ takes the input corresponding to the input for the adversary in the real world. Again, we denote the vector of all inputs with $\vec{x}$ (where again the entries in the vector corresponding to actively corrupted parties are fixed to $\square$). Over the adversarial interface to σ, $\mathcal{F}$ first forwards the inputs for passively corrupted parties to the adversary,[2] and then accepts inputs for actively corrupted parties. After that, $\mathcal{F}$ evaluates the circuit while communicating with σ according to its specification (i.e., depending on the guaranteed security properties) over the adversarial interface. At the end of the execution, $\mathcal{F}$ provides the output for each correct party over the corresponding interface. Furthermore, $\mathcal{F}$ provides the output for passively and actively corrupted parties over the adversarial interface to σ,[3] which subsequently outputs some function of its internal state. The vector of the output values of all parties (where, again, the output of actively corrupted parties is fixed to $\square$) together with the output of the simulator is denoted by $\text{IDEAL}^{\vec{x}}_{\mathcal{F},\sigma}$.

In both the real and the ideal world, the set of corrupted parties is the same and fixed before the beginning of the execution (*static corruption*). Furthermore, the ideal functionality, the adversary, and the simulator obtain this set as an input.

To formalize the statement that whatever the adversary can achieve in the real world, he can also achieve in the ideal world, the notion of a distinguisher is used: This distinguisher, denoted by D, chooses the input vector $\vec{x}$ and

[2] If secrecy is not guaranteed, $\mathcal{F}$ forwards the inputs of all parties.

[3] Again, if secrecy is not guaranteed, $\mathcal{F}$ reveals all outputs to σ.

obtains either $\text{REAL}^{\vec{x}}_{\pi,\mathcal{A}}$ or $\text{IDEAL}^{\vec{x}}_{\mathcal{F},\sigma}$. Then, D outputs a bit that, intuitively speaking, denotes which one of the two output collections it had obtained.

A protocol π is said to securely implement a functionality $\mathcal{F}$ if, for every possible set of corrupted parties and every adversary $\mathcal{A}$, there is a simulator σ such that for all distinguishers D the two executions $\text{REAL}^{\vec{x}}_{\pi,\mathcal{A}}$ and $\text{IDEAL}^{\vec{x}}_{\mathcal{F},\sigma}$ are indistinguishable, i.e.

$$|\text{Prob}[D(\text{IDEAL}^{\vec{x}}_{\mathcal{F},\sigma}) = 1] - \text{Prob}[D(\text{REAL}^{\vec{x}}_{\pi,\mathcal{A}}) = 1]| \leq \varepsilon$$

where ε denotes the error probability.

The choice of ε and the computing power of the involved entities depend on the desired level of security (perfect, statistical, or computational), and is discussed in the next section.

The security model as described here is called *stand-alone* security. It guarantees security only if a single protocol instance is executed at any point in time. In [Can01, BPW04] this model has been extended to allow for *universal composition* (UC). In the UC model, any number of protocols can be composed concurrently and/or sequentially, which allows for a modular analysis of protocols. For ease of presentation, we focus on stand-alone security. Yet, we conjecture that our results can also be carried over to the UC model.

While the paradigm of simulation based security allows for formally precise statements, proofs become very technical and most if not all of the intuition is lost. Therefore, in this thesis, we provide rather intuitive proofs which, formally speaking, are only proof sketches.

2.3.5 Levels of Security

Before defining the different levels of security, we need to introduce the notions of negligibility and efficiency. A *negligible* function is a function $f : \mathbb{N} \mapsto \mathbb{R}^+$ that goes faster to zero than the inverse of any polynomial. Formally, we say that $f(x)$ is negligible, if

$$\forall c > 0, \exists x_0, \forall x > x_0 : f(x) < \frac{1}{x^c}.$$

Closely related to the concept of negligibility is the concept of an overwhelming probability: A probability $p(\kappa)$ is *overwhelming* if $1 - p(\kappa)$ is negligible. Furthermore, we say a program is *efficient* if the number of steps it performs during the protocol execution is polynomial in the input size.

In the *information-theoretic* setting, both the adversary and the distinguisher are not restricted to efficient computation. This setting is further divided according to the tolerated error probability: In the *perfect setting*, the error probability is zero ($\varepsilon = 0$), whereas in the *statistical setting*, the error probability is a negligible function in the security parameter. That means that in the statistical setting, the adversary has a very small chance of breaking the security of the protocol, depending on the random choices, while in the perfect setting, security holds independent of the randomness.

In the *computational* setting (also called the cryptographic setting), security is based on some computational assumption, and only efficient adversaries and distinguishers are considered. Furthermore, as in the statistical setting, a negligible small error probability is tolerated (i.e., the error probability is a negligible function in the security parameter).

As a matter of fact, all programs π_i must be efficient in all settings. Furthermore, simulators must be efficient (in the runtime of the adversary) not only in the computational, but also in the information-theoretical setting, since otherwise, information-theoretical security does not imply computational security.

Chapter 3

Graceful Degradation

In this work, we introduce the notion of graceful degradation in several dimensions. More concretely, we provide protocols that provide both graceful degradation of security (hybrid security), as well as graceful degradation of corruptions (mixed adversaries). For this purpose, we introduce a notation to characterize the tolerated sets of corrupted parties, merging the notations for mixed adversaries (Section 2.2.3) and for hybrid security (Section 2.3.2).

3.1 General Adversaries

On the one hand, protocols for mixed general adversaries are traditionally characterized by an adversary structure $\mathcal{Z}$, which consists of tuples $(\mathcal{D}, \mathcal{E})$ of subsets of the player set, where $\mathcal{E}$ is the set of passively, and $\mathcal{D} \subseteq \mathcal{E}$ is the set of actively corrupted parties (see Section 2.2.3). On the other hand, each security guarantee depends on the sets of *actually* corrupted parties. We consider four security guarantees, namely correctness, secrecy, robustness, and fairness. Depending on the set of actually corrupted parties, different security properties are achieved. This is modeled with four adversary structures (see Section 2.3.2).

To characterize mixed general adversaries in a setting with hybrid security, we use four adversary structures, one for each security guarantee, and each consisting of tuples $(\mathcal{D}, \mathcal{E})$ specifying the tolerated combinations of sets of actively and passively corrupted parties, such that the security requirement is still guaranteed. More specifically, we consider the four adversary

structures $\mathcal{Z}^c$ for correctness, $\mathcal{Z}^s$ for secrecy, $\mathcal{Z}^r$ for robustness, and $\mathcal{Z}^f$ for fairness. Since all our protocols achieve agreement on abort for free, we do not introduce a separate structure for this security property.

Then, 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 $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^f$. Trivially, if several of these conditions are satisfied, all corresponding security properties are guaranteed. In particular, full security is guaranteed if the conditions for all four security properties are fulfilled.

We assume that $\mathcal{Z}^r \subseteq \mathcal{Z}^c$ and $\mathcal{Z}^s \subseteq \mathcal{Z}^c$, as secrecy and robustness are not well defined without correctness, and $\mathcal{Z}^f \subseteq \mathcal{Z}^s$ since in a setting without secrecy the adversary inherently has an unfair advantage over honest parties.

3.2 Threshold Adversaries

The characterization for general adversaries can be adjusted to threshold adversaries: We consider mixed adversaries which are characterized by two thresholds t_a and t_p, where up to t_p parties can be passively corrupted, and up to t_a of these parties can even be corrupted actively (see Section 2.2.3). Furthermore, the level of security depends on the number of *actually* corrupted parties; the fewer parties are corrupted, the more security is guaranteed. We consider four security properties, namely correctness, secrecy, robustness, and fairness, which is modeled with four thresholds, one for each security property.

To characterize mixed threshold adversaries in a setting with hybrid security, we use four pairs of thresholds, one for each security property, and each specifying the upper bound on the number of active and passive corruptions that the adversary may perform, such that the security property is still guaranteed. More specifically, we consider the four pairs of thresholds (t_a^c, t_p^c) for correctness, (t_a^s, t_p^s) for secrecy, (t_a^r, t_p^r) for robustness, and (t_a^f, t_p^f) for fairness. As in the setting with general adversaries, we do not introduce a separate pair of thresholds for agreement on abort. And again we assume that $(t_a^s, t_p^s) \leq (t_a^c, t_p^c)$ and $(t_a^r, t_p^r) \leq (t_a^c, t_p^c)$,[4] as secrecy and robustness are not well defined without correctness, and $(t_a^f, t_p^f) \leq (t_a^s, t_p^s)$, as fairness cannot be achieved without secrecy.

[4]We write $(t_a^s, t_p^s) \leq (t_a^c, t_p^c)$ as shorthand for $t_a^s \leq t_a^c$ and $t_p^s \leq t_p^c$.

Then, correctness is guaranteed for $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq (t_a^c, t_p^c)$, secrecy is guaranteed for $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq (t_a^s, t_p^s)$, robustness is guaranteed for $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq (t_a^r, t_p^r)$, and fairness is guaranteed for $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq (t_a^f, t_p^f)$. Trivially, if several of these conditions are satisfied, all corresponding security properties are guaranteed.

3.3 Multi-Threshold Adversaries

The traditional model for threshold mixed adversaries uses a single pair of thresholds (t_a, t_p), where the protocol tolerates up to t_p passive corruptions, of which up to t_a may even be active. This model allows to define only a single maximal adversary. As an example, consider the following two pairs of thresholds (t_a, t_p) and (t'_a, t'_p) with $t_a < t'_a$ and $t_p > t'_p$. Now, assume a setting where security against both (t_a, t_p) and (t'_a, t'_p), but not against (t'_a, t_p), can be guaranteed by a single protocol. That means, the protocol is secure against two incomparable maximal adversaries, but not against their "union". In the traditional model, it is impossible to express this bound, and one has to choose either one of the two maximal adversaries.

To model security guarantees against two or more incomparable maximal adversaries, we consider multiple pairs of thresholds. Therefore, we use multi-thresholds $T = \{(t_{a,1}, t_{p,1}), \ldots, (t_{a,k}, t_{p,k})\}$, i.e., sets of pairs of thresholds (t_a, t_p). In this model, security is guaranteed if $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq (t_a, t_p)$ for some $(t_a, t_p) \in T$, denoted by $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq T$.

Again, the guaranteed security properties (correctness, secrecy, robustness, and fairness) depend on the number $(|\mathcal{D}^*|, |\mathcal{E}^*|)$ of *actually* corrupted parties (hybrid security). As before, we do not introduce a separate pair of thresholds for agreement on abort. Hence, we consider the four multi-thresholds T^c, T^s, T^r, and T^f: Correctness is guaranteed for $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq T^c$, 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 $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq T^f$. As for general and threshold adversaries, if several of these conditions are satisfied, all corresponding security properties are guaranteed. Also, we have again the assumption that $T^s \leq T^c$ and $T^r \leq T^r$,[5] as secrecy and robustness are not well defined without correctness, and $T^f \leq T^s$ as fairness cannot be achieved without secrecy.

Note that incomparable adversaries can only exist if both the set of active and passive corruptions are relevant for a security property. If each of the

[5] We write $T_1 \leq T_2$ as a shorthand for $\forall (t_a, t_p) \in T_1, \exists (t'_a, t'_p) \in T_2 : (t_a, t_p) \leq (t'_a, t'_p)$.

security properties is influenced only by one of the two sets, the model with simple thresholds is sufficiently expressive.

Chapter 4

Protocols with Perfect Security

We first present the protocols for the setting with perfect security. The results in this setting are rather straight-forward and less complex than in the other settings. This allows us to present our general approach: We design parametrized protocols and analyze when the various security properties are guaranteed. Then, given the security of the parametrized protocols, we deduce bounds when reactive MPC can be achieved, and prove their tightness.

For the construction of the perfectly secure protocols, we parametrize existing protocols ([Mau02] for general, and [BGW88] for threshold adversaries). On an abstract level, our modifications can be described as follows: First, we define the state that is held in the protocol in terms of a parameter that influences the secrecy. Second, given the parameter for secrecy, we express the reconstruct protocol in terms of an additional parameter determining the amount of error correction taking place without aborting the protocol, i.e., a parameter for robustness. Traditional protocols correct as many errors as possible. By using a parameter, our protocol may stay below the theoretical limit, thereby providing extended error detection. To our knowledge, such a second parameter has not been considered before. Then, we analyze the parametrized protocols with respect to correctness, secrecy, and robustness (fairness is treated separately).

The treatment follows along the lines of [HLMR11], where this setting was first discussed.

4.1 General Adversaries

The solution for the setting with general adversaries is conceptually simpler and provides more insight into the underlying problem. Therefore, we discuss this setting first.

4.1.1 A Parametrized Protocol for General Adversaries

As discussed above, we generalize the protocol of [Mau02] introducing two parameters: The *sharing specification* $\mathcal{S} = (S_1, \ldots, S_\ell)$ is a parameter of the state and influences the secrecy. The *robustness parameter* $\mathcal{R} = \{R_1, \ldots, R_k\}$ is a parameter of the reconstruction protocol and determines the amount of error correction: If the errors can be explained with a set $R \in \mathcal{R}$ of actively corrupted parties, the errors are ignored (corrected) and the protocol proceeds. Otherwise, the protocol is aborted. Both parameters are collections of subsets of $\mathcal{P}$.

4.1.1.1 The Underlying Verifiable Secret Sharing Scheme

The state of the protocol is maintained with an ℓ-out-of-ℓ sharing, where each party holds several summands.

Definition 1 ($\mathcal{S}$-Sharing)**.** A value s is *$\mathcal{S}$-shared* for sharing specification $\mathcal{S} = (S_1, \ldots, S_\ell)$ if there are values $s_1, \ldots, s_\ell \in \mathbb{F}$, such that $s_1 + \ldots + s_\ell = s$ and, for all i, every (correct) party $p_j \in S_i$ holds the summand s_i. A sharing specification $\mathcal{S}$ is *$\mathcal{E}$-secret* if the summands held by the parties in $\mathcal{E}$ are statistically independent from the secret, and *$\mathcal{D}$-permissive* if the summands held by the parties in $\mathcal{P} \setminus \mathcal{D}$ uniquely define the secret.

Lemma 1. *An $\mathcal{S}$-sharing is $\mathcal{E}$-secret if $\exists S_i \in \mathcal{S} : S_i \cap \mathcal{E} = \emptyset$, and $\mathcal{D}$-permissive if $\forall S_i \in \mathcal{S} : S_i \setminus \mathcal{D} \neq \emptyset$.*

Proof. Secrecy follows from the fact that $\mathcal{E}$ lacks at least one summand s_i. Furthermore, given that $\forall S_i \in \mathcal{S} : S_i \setminus \mathcal{D} \neq \emptyset$, each summand s_i is held by at least one party in $\mathcal{P} \setminus \mathcal{D}$. 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 the secret s (see Figure 4.1).

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

1. The dealer chooses uniformly random summands $s_1, \ldots, s_\ell$ such that $s = \sum_{i=1}^{\ell} s_i$, where $\ell = |\mathcal{S}|$. Then, for $i = 1, \ldots, \ell$, 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. The dealer broadcasts each summand s_i for which inconsistencies were reported, and the players in S_i accept this summand.
4. Each party p_j outputs its share $\{s_i \mid p_j \in S_i\}$.

Figure 4.1: The perfectly secure share protocol for general adversaries.

Lemma 2. *Let $\mathcal{S}$ be the sharing specification. On input s from the dealer, $\text{SHARE}_{\text{P}}^{\text{GA}}$ 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, a valid $\mathcal{S}$-sharing is distributed in the first step. 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 with $p_j \in S_i$ reported an inconsistency. Yet, such an inconsistency occurs only if one of the parties in S_i is actively corrupted, i.e., when the adversary knew the value already beforehand.

CORRECTNESS: First, we have to show that the protocol outputs a valid $\mathcal{S}$-sharing. Due to the bilateral checks, the summands held by correct parties are always consistent, which implies already 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 cannot be aborted. □

For the public reconstruction[6] of a shared value, we modify the reconstruction protocol of [Mau02]. In our protocol, we trade correctness for robustness by introducing a robustness parameter $\mathcal{R}$. First, each summand s_i is broadcasted by all parties in S_i. Then, if the inconsistencies can be explained with a faulty set $R \in \mathcal{R}$, the values from parties in R are ignored (corrected), and reconstruction proceeds. Otherwise, the protocol is aborted. PUBLIC RECONSTRUCTION$_{\mathrm{P}}^{\mathrm{GA}}$ is the only subprotocol that might abort. All other protocols abort only if they use PUBLIC RECONSTRUCTION$_{\mathrm{P}}^{\mathrm{GA}}$ as a subprotocol and the invocation thereof aborts. Therefore, it is sufficient to discuss agreement on abort only for this protocol.

Whenever two sets of possibly actively corrupted parties cover a set $S \in \mathcal{S}$, i.e. $S \subseteq \mathcal{D}_1 \cup \mathcal{D}_2$, and the parties in $\mathcal{D}_1$ contradict the parties in $\mathcal{D}_2$, then it is impossible to decide which is the correct value. This observation implies an upper bound on $\mathcal{R}$, namely $\forall S \in \mathcal{S}, R_1, R_2 \in \mathcal{R} : S \not\subseteq R_1 \cup R_2$. However, instead of always correcting as many errors as possible, the protocol allows to select a parameter $\mathcal{R}$ that remains below this upper bound (i.e. contains smaller sets R). Now, when only correcting errors that are covered by a set $R \in \mathcal{R}$, then we can detect errors in sets $\mathcal{D}$ where $\forall S \in \mathcal{S}, R \in \mathcal{R} : S \not\subseteq \mathcal{D} \cup R$. Hence, this approach provides a tradeoff between reduced robustness and extended correctness.

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

1. For each summand s_i:
   (a) Each party $p_j \in S_i$ broadcasts s_i. For $p_j \in S_i$, let $s_i^{(j)}$ denote the value (for s_i) broadcasted by party p_j.
   (b) Each party (locally) reconstructs the summand s_i: If there is a value s_i such that there exists $R \in \mathcal{R}$ with $s_i^{(j)} = s_i$ for all $p_j \in S_i \setminus R$, use s_i. Otherwise abort.
2. Each party outputs the secret $s = s_1 + \ldots + s_\ell$.

Figure 4.2: The perfectly secure public reconstruction protocol for general adversaries.

[6] Private reconstruction can be reduced to public reconstruction using a blinding technique as discussed in Section 2.3.3. Note that the trivial solution, where each party sends its share to party p_k, does not achieve agreement on abort.

Lemma 3. *Let $\mathcal{S}$ be the sharing specification, and $\mathcal{R}$ be the robustness parameter, where $\forall S \in \mathcal{S}, R_1, R_2 \in \mathcal{R} : S \not\subseteq R_1 \cup R_2$. Given an $\mathcal{S}$-sharing of some value s,* PUBLIC RECONSTRUCTION$_{\mathrm{P}}^{\mathrm{GA}}$ *is correct if $\forall R \in \mathcal{R}, S \in \mathcal{S} : S \setminus R \not\subseteq \mathcal{D}^*$, is robust if $\mathcal{D}^* \in \mathcal{R}$, and always guarantees agreement on abort.*

Proof. CORRECTNESS: The condition $\forall R \in \mathcal{R}, S \in \mathcal{S} : S \setminus R \not\subseteq \mathcal{D}^*$ states that for every summand s_i and every set $R \in \mathcal{R}$, there is at least one correct party whose summand is not ignored. Hence, if a value s_i is chosen, it must be the correct one.

ROBUSTNESS: When reconstructing the summand s_i, all but the actively corrupted parties in $\mathcal{D}^*$ broadcast the same summand s_i. Since $\mathcal{D}^* \in \mathcal{R}$, these inconsistencies can be explained with a set in $\mathcal{R}$. Hence, the corresponding set can be ignored and reconstruction terminates without abort.

Since the abort decision is based only on broadcasted values, we always have AGREEMENT ON ABORT (cf. Section 2.3.2). □

4.1.1.2 Addition, Multiplication, and Random Values

Linear functions (and in particular additions) can be computed locally, since $\mathcal{S}$-sharings are linear. In particular, given sharings of a and b, and a constant c, one can easily compute the 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 $r = r_1 + \ldots + r_n$ (cf. Section 2.3.3).

For the multiplication of two values a and b, we use the protocol from [Mau02], based on our modified share and reconstruct protocols. The multiplication protocol exploits the fact that $ab = \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} a_i b_j$: For each $a_i b_j$, first, all parties who know a_i and b_j compute $a_i b_j$ and share it. Then, all parties choose a (correct) sharing of $a_i b_j$. In the end, each party locally computes the linear function described above. In order to ensure the correctness of the sharing of $a_i b_j$, the protocol checks whether all parties that computed $a_i b_j$ shared the same value. If this holds, and if at least one correct party shared $a_i b_j$, all sharings contain the correct value, and an arbitrary one can be chosen. Otherwise, at least one of the parties that computed $a_i b_j$ is actively corrupted, and the summands a_i and b_j can be reconstructed without violating secrecy.

Protocol MULTIPLICATION$_{\mathrm{P}}^{\mathrm{GA}}$: Given $\mathcal{S}$-sharings of some values $a = \sum_{i=1}^{\ell} a_i$ and $b = \sum_{i=1}^{\ell} b_i$, compute an $\mathcal{S}$-sharing of the product $c = ab$.

1. For each pair $S_i, S_j \in \mathcal{S}$:
   (a) Each party in $S_i \cap S_j$ computes $a_i b_j$ and invokes SHARE$_{\mathrm{P}}^{\mathrm{GA}}$ on it.
   (b) All parties (distributedly) compute the difference of the value shared by the party with the smallest index in $S_i \cap S_j$ and each other party in $S_i \cap S_j$, and invoke PUBLIC RECONSTRUCTION$_{\mathrm{P}}^{\mathrm{GA}}$ on it.
   (c) If all these opened differences are 0, then the sharing by the party with the smallest index in $S_i \cap S_j$ is used as the sharing of $a_i b_j$. Otherwise, a_i and b_j are reconstructed along the lines of PUBLIC RECONSTRUCTION$_{\mathrm{P}}^{\mathrm{GA}}$, and a default sharing of $a_i b_j$ is used.
2. The parties (distributedly) compute the sum of their sharings of all terms $a_i b_j$, resulting in a sharing of $c = ab$.

Figure 4.3: The perfectly secure multiplication protocol for general adversaries.

Lemma 4. *Let $\mathcal{S}$ be the sharing and $\mathcal{R}$ be the robustness parameter, where $\forall S \in \mathcal{S}, R_1, R_2 \in \mathcal{R} : S \not\subseteq R_1 \cup R_2$. Given $\mathcal{S}$-sharings of a and b,* MULTIPLICATION *outputs a correct $\mathcal{S}$-sharing of the product $c = ab$ if $\forall S_i, S_j \in \mathcal{S} : S_i \cap S_j \not\subseteq \mathcal{D}^*$ and the subprotocols are correct,[7] is secret if the subprotocols are secret and correct, and is robust if the subprotocols are robust.*

Proof. SECRECY: Secrecy is guaranteed if the adversary cannot provoke the reconstruction of summands a_i and b_j where he did not know a_i or b_j beforehand, i.e., where $S_i \cap \mathcal{D}^* = \emptyset$ or $S_j \cap \mathcal{D}^* = \emptyset$. In that case, we have in particular that $(S_i \cap S_j) \cap \mathcal{D}^* = \emptyset$, i.e., only correct parties share the product $a_i b_j$ (Step 1). Hence, given correct reconstruction (Step 2), all differences in Step 3 are 0, and the summands are not reconstructed.

CORRECTNESS: Correctness for the summand of the product $a_i b_j$ is guaranteed given that reconstruction is correct, and if there is at least one correct

[7] In particular $\forall S_i, S_j \in \mathcal{S} : S_i \cap S_j \neq \emptyset$, since otherwise no party can compute $a_i b_j$.

party in $S_i \cap S_j$ that correctly computes and shares this value, which is guaranteed by the premise in the lemma.

ROBUSTNESS: This security requirement follows directly from the robustness of PUBLIC RECONSTRUCTION$_{\mathrm{P}}^{\mathrm{GA}}$ and SHARE$_{\mathrm{P}}^{\mathrm{GA}}$. □

4.1.1.3 The Security of the Generalized Protocol from [Mau02]

Considering the security of the subprotocols described above, we can derive the security of the parametrized protocol, denoted by $\pi_{\mathrm{P}}^{\mathcal{S},\mathcal{R}}$:

Lemma 5. *Let $\mathcal{S}$ be the sharing specification, and $\mathcal{R}$ be the robustness parameter, where $\forall S \in \mathcal{S}, R_1, R_2 \in \mathcal{R} : S \not\subseteq R_1 \cup R_2$. The protocol $\pi_{\mathrm{P}}^{\mathcal{S},\mathcal{R}}$ guarantees*

- CORRECTNESS *if* $\forall S_i, S_j \in \mathcal{S} : S_i \cap S_j \not\subseteq \mathcal{D}^*$ *and* $\forall R \in \mathcal{R}, S \in \mathcal{S} : S \setminus R \not\subseteq \mathcal{D}^*$,
- SECRECY *if* $\exists S_i \in \mathcal{S} : S_i \cap \mathcal{E}^* = \emptyset$ *and correctness is guaranteed,*
- ROBUSTNESS *if* $\mathcal{D}^* \in \mathcal{R}$, *and*
- AGREEMENT ON ABORT *always.*

Proof. For each property, given its condition in the lemma, it is easy to verify that the corresponding conditions for this property in all subprotocols are fulfilled. □

4.1.2 A Trivial Non-Secret Protocol

If there is no secrecy requirement (i.e., if $\mathcal{Z}^s = \{(\emptyset, \emptyset)\}$), each party can simply broadcast its inputs, and all parties locally compute the output.

Trivially, this protocol achieves correctness and robustness for any number of corrupted parties. As mentioned in Section 2.3.2, in a setting without secrecy, we do not achieve the traditional interpretation of correctness: We can obtain neither input-independence nor true randomness. Yet, the output is guaranteed to be consistent with the input of the correct parties.

Protocol for non-secret MPC: Given a circuit, evaluate the circuit gate-wise.

- Input gates: Party p_i provides input s by broadcasting s to all parties.
- Addition and multiplication gates: Each party locally performs all computations on the broadcasted values.
- Random gates: Party p_1 broadcasts a random value r.
- Output gates: Each party outputs the value computed locally before.

Figure 4.4: A protocol without secrecy.

4.1.3 The Main Result for General Adversaries

The following theorem states the optimal bound for perfectly secure reactive MPC with graceful degradation of both security (allowing for hybrid security) and corruptions (allowing for mixed adversaries) for general adversaries, given broadcast. Furthermore, we show that the bound is sufficient for MPC by providing parameters for the generalized protocol described in Section 4.1.1. In the following section, we prove that the bound is also necessary.

Theorem 1. *In the secure channels model with broadcast and general adversaries, perfectly 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*

$$\big(\forall(\mathcal{D}^c,\cdot) \in \mathcal{Z}^c, (\cdot,\mathcal{E}_1^s), (\cdot,\mathcal{E}_2^s) \in \mathcal{Z}^s, (\mathcal{D}^r,\cdot) \in \mathcal{Z}^r :$$
$$\mathcal{D}^c \cup \mathcal{E}_1^s \cup \mathcal{D}^r \neq \mathcal{P} \quad \wedge \quad \mathcal{D}^c \cup \mathcal{E}_1^s \cup \mathcal{E}_2^s \neq \mathcal{P}\big)$$
$$\vee \quad \mathcal{Z}^s = \{(\emptyset,\emptyset)\}.$$

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

Note that the bound holds for any $\mathcal{Z}^f \subseteq \mathcal{Z}^s$, and we can always have $\mathcal{Z}^f = \mathcal{Z}^s$.

Proof. The necessity of the bound in the theorem is proven in Section 4.1.4. To prove that the bound is also sufficient, we first note that if $\mathcal{Z}^s = \{(\emptyset,\emptyset)\}$, then there is no secrecy requirement, and we can directly use the trivial non-secret protocol described in Section 4.1.2. Otherwise, we employ the parametrized

version $\pi_{\mathrm{P}}^{\mathcal{S},\mathcal{R}}$ of the protocol of [Mau02] described in Section 4.1.1. We set $\mathcal{S} := \{\overline{\mathcal{E}^s} \mid (\cdot, \mathcal{E}^s) \in \mathcal{Z}^s\}$ and $\mathcal{R} = \{\mathcal{D} \mid (\mathcal{D}, \cdot) \in \mathcal{Z}^r \cup \mathcal{Z}^f\}$.

The precondition of Lemma 5 is that $\forall S \in \mathcal{S}, R_1, R_2 \in \mathcal{R} : S \not\subseteq R_1 \cup R_2$, which is equivalent to $\forall S \in \mathcal{S}, R_1, R_2 \in \mathcal{R} : \overline{S} \cup R_1 \cup R_2 \neq \mathcal{P}$. We have that $\mathcal{Z}^r \cup \mathcal{Z}^f \subseteq \mathcal{Z}^c$. Hence, for our choice of parameters $\mathcal{S}$ and $\mathcal{R}$, we have that for all S, R_1, and R_2, there are sets $\mathcal{E}^s$ and $\mathcal{D}^c$, such that $\overline{S} \cup R_1 \cup R_2 \subseteq \mathcal{E}^s \cup \mathcal{D}^c \cup R_2$. By construction of $\mathcal{R}$, $(R_2, \cdot)$ is either from $\mathcal{Z}^r$ or from $\mathcal{Z}^f$. Therefore, the inequality follows either from the left-hand side (if $(R_2, \cdot) \in \mathcal{Z}^r$) or from the right-hand side (if $(R_2, \cdot) \in \mathcal{Z}^f \subseteq \mathcal{Z}^s$) of the condition in the theorem. Hence, we can apply the lemma 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.

For fairness we have $\forall(\mathcal{D}^f, \cdot) \in \mathcal{Z}^f : \mathcal{D}^f \in \mathcal{R}$. Hence, for $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^f$ the protocol is robust, and the adversary cannot abort. □

4.1.4 Proofs of Necessity

In this section, we prove that the bound in Theorem 1 is necessary, i.e., if violated, some (reactive) functionalities cannot be securely computed. The bound is violated if

$$\begin{aligned}&\big(\exists(\mathcal{D}^c, \cdot) \in \mathcal{Z}^c, (\cdot, \mathcal{E}_1^s), (\cdot, \mathcal{E}_2^s) \in \mathcal{Z}^s, (\mathcal{D}^r, \cdot) \in \mathcal{Z}^r : \\ &\qquad\qquad \mathcal{D}^c \cup \mathcal{E}_1^s \cup \mathcal{D}^r = \mathcal{P} \ \vee \ \mathcal{D}^c \cup \mathcal{E}_1^s \cup \mathcal{E}_2^s = \mathcal{P}\big) \\ \wedge \quad &\mathcal{Z}^s \neq \{(\emptyset, \emptyset)\}.\end{aligned}$$

Due to monotonicity, we can assume that the sets $\mathcal{D}^c$, $\mathcal{E}_1^s$, $\mathcal{E}_2^s$, and $\mathcal{D}^r$ are disjoint. Furthermore, since $\mathcal{Z}^s \neq \{(\emptyset, \emptyset)\}$, we can assume that $\mathcal{E}_1^s \neq \emptyset$. We can split the condition according to whether $\mathcal{D}^c \cup \mathcal{E}_1^s \cup \mathcal{D}^r = \mathcal{P}$ or $\mathcal{D}^c \cup \mathcal{E}_1^s \cup \mathcal{E}_2^s = P$.

1. $\exists(\mathcal{D}^c, \cdot) \in \mathcal{Z}^c, (\cdot, \mathcal{E}_1^s) \in \mathcal{Z}^s, (\mathcal{D}^r, \cdot) \in \mathcal{Z}^r : \mathcal{D}^c \cup \mathcal{E}_1^s \cup \mathcal{D}^r = \mathcal{P} \ \wedge \ \mathcal{E}_1^s \neq \emptyset$. We further split this case according to whether $\mathcal{D}^c = \emptyset$ or $\mathcal{D}^r = \emptyset$. Since $\mathcal{Z}^r \subseteq \mathcal{Z}^c$, the case where $\mathcal{D}^c = \emptyset \wedge \mathcal{D}^r \neq \emptyset$ is subsumed by Case 1(b).
   (a) $\exists(\mathcal{D}^c, \cdot) \in \mathcal{Z}^c, (\cdot, \mathcal{E}_1^s) \in \mathcal{Z}^s, (\mathcal{D}^r, \cdot) \in \mathcal{Z}^r :$ $\mathcal{D}^c \cup \mathcal{E}_1^s \cup \mathcal{D}^r = \mathcal{P} \ \wedge \ \mathcal{E}_1^s \neq \emptyset \wedge \mathcal{D}^c \neq \emptyset \wedge \mathcal{D}^r \neq \emptyset$ (Section 4.1.4.1)
   (b) $\exists(\mathcal{D}^c, \cdot) \in \mathcal{Z}^c, (\cdot, \mathcal{E}_1^s) \in \mathcal{Z}^s : \mathcal{D}^c \cup \mathcal{E}_1^s = \mathcal{P} \ \wedge \ \mathcal{E}_1^s \neq \emptyset \wedge \mathcal{D}^c \neq \emptyset$ (Section 4.1.4.2)

(c) $\exists(\cdot, \mathcal{E}_1^s) \in \mathcal{Z}^s : \mathcal{E}_1^s = \mathcal{P} \wedge \mathcal{E}_1^s \neq \emptyset$: Due to monotonicity and $|\mathcal{P}| = n \geq 2$, this implies $\exists(\cdot, \mathcal{E}_1^s), (\cdot, \mathcal{E}_2^s) \in \mathcal{Z}^s :$
$\mathcal{E}_1^s \cup \mathcal{E}_2^s = \mathcal{P} \wedge \mathcal{E}_1^s \neq \emptyset \wedge \mathcal{E}_2^s \neq \emptyset \wedge \mathcal{E}_1^s \cap \mathcal{E}_2^s = \emptyset$,
which is identical to Case 2(b).

2. $\exists(\mathcal{D}^c, \cdot) \in \mathcal{Z}^c, (\cdot, \mathcal{E}_1^s), (\cdot, \mathcal{E}_2^s) \in \mathcal{Z}^s : \mathcal{D}^c \cup \mathcal{E}_1^s \cup \mathcal{E}_2^s = \mathcal{P} \wedge \mathcal{E}_1^s \neq \emptyset$. Again, we further split this case according to whether $\mathcal{D}^c = \emptyset$ or $\mathcal{E}_2^s = \emptyset$. Note that the case where $\mathcal{D}^c \neq \emptyset \wedge \mathcal{E}_2^s = \emptyset$ is identical to Case 1(b), and the case where $\mathcal{D}^c = \emptyset \wedge \mathcal{E}_2^s = \emptyset$ is identical to Case 1(c).

   (a) $\exists(\mathcal{D}^c, \cdot) \in \mathcal{Z}^c, (\cdot, \mathcal{E}_1^s), (\cdot, \mathcal{E}_2^s) \in \mathcal{Z}^s :$
   $\mathcal{D}^c \cup \mathcal{E}_1^s \cup \mathcal{E}_2^s = \mathcal{P} \wedge \mathcal{E}_1^s \neq \emptyset \wedge \mathcal{E}_2^s \neq \emptyset \wedge \mathcal{D}^c \neq \emptyset$ (Section 4.1.4.3)

   (b) $\exists(\cdot, \mathcal{E}_1^s), (\cdot, \mathcal{E}_2^s) \in \mathcal{Z}^s : \mathcal{E}_1^s \cup \mathcal{E}_2^s = \mathcal{P} \wedge \mathcal{E}_1^s \neq \emptyset \wedge \mathcal{E}_2^s \neq \emptyset$ (Section 4.1.4.4)

4.1.4.1 Case 1(a): $\exists(\mathcal{D}^c, \cdot) \in \mathcal{Z}^c, (\cdot, \mathcal{E}_1^s) \in \mathcal{Z}^s, (\mathcal{D}^r, \cdot) \in \mathcal{Z}^r :$ $\mathcal{D}^c \cup \mathcal{E}_1^s \cup \mathcal{D}^r = \mathcal{P} \wedge \mathcal{E}_1^s \neq \emptyset \wedge \mathcal{D}^c \neq \emptyset \wedge \mathcal{D}^r \neq \emptyset$

A state is a requirement for reactive functionalities. We first prove that it is impossible to hold a state in a specific 3-party setting. This proof is inspired by [BFH+08].

Definition 2 (State)**.** A *state* for n parties $p_1, \ldots, p_n$ is a tuple $(s_1, \ldots, s_n)$ that defines a value $s \in \{0, 1, \perp\}$, where party p_i holds s_i. A state is *secret* if the state information held by corrupted parties contains no information about the bit s. A state is *correct* if it uniquely defines either s or $\perp$. A state is *robust* if it uniquely defines either 0 or 1.

Lemma 6. *Three parties A, B, and C cannot hold a state (s_A, s_B, s_C) that defines a bit s providing secrecy in case of a passively corrupted A, correctness and robustness in case of an actively corrupted B, and correctness (without agreement on abort) in case of an actively corrupted C.*

Proof. To arrive at a contradiction, assume that (a, b, c) is a state for $s = 0$. Due to secrecy in case of a passively corrupted A, there exists b' and c' such that (a, b', c') is a valid state for $s = 1$. Due to correctness and robustness in case of an actively corrupted B, the state $(a, \cdot, c)$ must define the value 0 (where $\cdot$ is a placeholder for an arbitrary state information held by B). Due to correctness in case of an actively corrupted C, the state $(a, b', \cdot)$ defines either 1 or $\perp$. As a consequence, with probability greater 0, the state (a, b', c) can be achieved if $s = 0$ and B is actively corrupted, and it can be achieved if $s = 1$ and C is actively corrupted. Hence, it must define both 0 and either 1 or $\perp$, which is a contradiction. □

Given Lemma 6, we can prove the desired bound by reducing the n-party setting to the 3-party setting specified there: Assume we have a perfectly secure n-party state $(s_1, \ldots, s_n)$ for the case $\exists(\mathcal{D}^c, \cdot) \in \mathcal{Z}^c, (\cdot, \mathcal{E}_1^s) \in \mathcal{Z}^s, (\mathcal{D}^r, \cdot) \in \mathcal{Z}^r : \mathcal{D}^c \cup \mathcal{E}_1^s \cup \mathcal{D}^r = \mathcal{P} \wedge \mathcal{E}_1^s \neq \emptyset \wedge \mathcal{D}^c \neq \emptyset \wedge \mathcal{D}^r \neq \emptyset$. By assumption we have that $\mathcal{D}^c$, $\mathcal{E}_1^s$, and $\mathcal{D}^r$ are disjoint.

We obtain a 3-party state (s_A, s_B, s_C) from $(s_1, \ldots, s_n)$ by having A, B, and C emulate the parties in $\mathcal{E}_1^s$, $\mathcal{D}^r$, and $\mathcal{D}^c$ respectively. The state $(s_1, \ldots, s_n)$ tolerates passive corruption of all parties in $\mathcal{E}_1^s$ while maintaining secrecy, active corruption of all parties in $\mathcal{D}^r$ while maintaining correctness and robustness, and active corruption of all parties in $\mathcal{D}^c$ while maintaining correctness. Hence, the resulting state (s_A, s_B, s_C) is secure for the specific corruption setting specified in Lemma 6, which is a contradiction.

4.1.4.2 Case 1(b): $\exists(\mathcal{D}^c, \cdot) \in \mathcal{Z}^c, (\cdot, \mathcal{E}_1^s) \in \mathcal{Z}^s : \mathcal{D}^c \cup \mathcal{E}_1^s = \mathcal{P} \ \wedge \ \mathcal{E}_1^s \neq \emptyset \wedge \mathcal{D}^c \neq \emptyset$

Analogously to the previous section, we prove that it is impossible to hold a state in a specific 2-party setting:

Lemma 7. *Two parties A and B cannot hold a state (s_A, s_B) that defines a bit s providing secrecy in case of a passively corrupted A, and correctness in case of an actively corrupted B.*

Proof. For a contradiction, assume that (a, b) is a state for $s = 0$. Due to secrecy in case of a passively corrupted A, there exists b' such that (a, b') is a valid state for $s = 1$. As a consequence, with probability greater 0, an actively corrupted B can choose between the state (a, b) and (a, b'), violating correctness. □

Given Lemma 7, we can prove the desired bound by reducing the n-party setting to the 2-party setting along the lines of the previous case.

4.1.4.3 Case 2(a): $\exists(\mathcal{D}^c, \cdot) \in \mathcal{Z}^c, (\cdot, \mathcal{E}_1^s), (\cdot, \mathcal{E}_2^s) \in \mathcal{Z}^s : \mathcal{D}^c \cup \mathcal{E}_1^s \cup \mathcal{E}_2^s = \mathcal{P} \ \wedge \ \mathcal{E}_1^s \neq \emptyset \wedge \mathcal{E}_2^s \neq \emptyset \wedge \mathcal{D}^c \neq \emptyset$

We first prove impossibility of computing the logical "and" in a specific 3-party setting.

Lemma 8. *Consider protocols for three parties A (with input $a \in \{0,1\}$), B (with input $b \in \{0,1\}$), and C (without input) that compute the logical "and" $z = a \wedge b$ and output it to all parties. There is no such protocol providing secrecy when A or B are passively corrupted, and correctness when C is actively corrupted.*

Proof. To arrive at a contradiction, assume that a secure protocol exists. We consider the random variables T_{AB}, T_{AC} and T_{BC} describing the transcripts of the channels connecting parties A and B, A and C, and B and C, respectively, and T describing the transcript of the broadcast channel, for honest protocol executions.

First, observe that for $a = 0$, we have $z = 0$ independent of b, hence $I(b; T_{AB}, T_{AC}, T | a = 0) = 0$. Analogously, for $a = 1$, A must learn $z = b$, hence $H(b|T_{AB}, T_{AC}, T, a = 1) = 0$. We distinguish two cases, namely when $H(b|T_{AB}, T, a = 1)$ is zero (i) or non-zero (ii).

In case (i), it follows from $I(b; T_{AB}, T_{AC}, T|a = 0) = 0$, that in particular we must have $I(b; T_{AB}, T|a = 0) = H(b|a = 0) - H(b|T_{AB}, T, a = 0) = 0$, and hence $H(b|T_{AB}, T, a = 0) = H(b|a = 0) > 0$. Furthermore, by assumption we have $H(b|T_{AB}, T, a = 1) = 0$. That means that party B can decide if $a = 0$ or $a = 1$ by observing the transcripts T_{AB} and T. This contradicts the secrecy in presence of a passively corrupted party B.

In case (ii), let $(t_{AB}, t_{AC}, t_{BC}, t)$ be a list of transcripts corresponding to a protocol run with $a = 1$ and $b = 0$. It follows from $H(b|T_{AB}, T, a = 1) > 0$ that there are transcripts t'_{AC} and t'_{BC}, such that $(t_{AB}, t'_{AC}, t'_{BC}, t)$ is a list of transcripts corresponding to a protocol run with $a = 1$ and $b = 1$. Thus, when observing t_{AB}, t'_{AC}, and t, party A cannot distinguish whether $b = 1$ and all parties behave correctly, or whether $b = 0$ and party C is actively corrupted provoking a wrong transcript t'_{AC} (which C achieves with non-zero probability). In the first scenario, due to completeness, A must output 1. In the second scenario, due to correctness, party A must output 0 (or abort). This is a contradiction. □

Given Lemma 8, we can prove the desired bound by reducing the n-party setting to the 3-party setting along the lines of the previous cases.

4.1.4.4 Case 2(b): $\exists(\cdot, \mathcal{E}_1^s), (\cdot, \mathcal{E}_2^s) \in \mathcal{Z}^s : \mathcal{E}_1^s \cup \mathcal{E}_2^s = \mathcal{P} \ \wedge \ \mathcal{E}_1^s \neq \emptyset \wedge \mathcal{E}_2^s \neq \emptyset$

As stated in [BGW88, Kil00], it is impossible to compute the logical "and" with perfect secrecy in a 2-party setting. Again, we can prove the desired

bound by reducing the n-party setting to the 2-party setting along the lines of the previous cases.

4.2 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 super-polynomial 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.

It follows from the bound for general adversaries in Theorem 1 that in the perfect setting, incomparable maximal adversaries cannot occur. Therefore, the adversary model with simple thresholds as introduced in Section 3.2 is sufficient and we do not need to use multi-thresholds.

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.

4.2.1 A Parametrized Protocol for Threshold Adversaries

Analogously to the setting with general adversaries, we generalize the perfectly secure MPC protocol of [BGW88] by introducing two parameters: On the one hand, the parameter of the state that influences the secrecy is the degree d of the sharing polynomial (see also [FHM98]). On the other hand, the robustness parameter of the reconstruct protocol that determines the amount of error correction taking place (without aborting the protocol) is the number e of corrected errors during reconstruction. The two parameters must fulfill $d + 2e < n$. Note that by staying below the theoretical limit and choosing $d + 2e \neq n - 1$, it is possible to reduce robustness for extended correctness. In [BGW88], both parameters are set to $d = e = t$, the maximum number of actively corrupted parties.

In the following, we present the parametrized protocol and analyze it with respect to correctness, secrecy, and robustness. We do not consider fairness. The main result (including fairness) is discussed in Section 4.2.2.

4.2.1.1 The Underlying Verifiable Secret Sharing Scheme

The state of the protocol is maintained with a Shamir sharing [Sha79] of each value.

Definition 3 (d-Sharing)**.** A value s is *d-shared* when there is a share polynomial $\hat{s}(x)$ of degree d with $\hat{s}(0) = s$, and every (correct) party p_i holds a share $s_i = \hat{s}(\alpha_i)$. We denote a d-sharing of s with $[s]$, and the share s_i also with $[s]_i$. A sharing degree d is *t_p-secret* if the shares held by the parties in any set of size at most t_p are statistically independent from the secret, and *t_a-permissive* if the shares of all but t_a parties uniquely define the secret.

Lemma 9. *A d-sharing is t_p-secret if $t_p \leq d$, and t_a-permissive if $t_a < n - d$.*

Proof. It follows directly from the properties of a polynomial of degree d that any set of at most d parties has no information about the secret. Furthermore, $t_a < n-d$ implies that there remain at least $d+1$ 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 Figure 4.5).

Lemma 10. *Let d be the sharing degree. On input s from the dealer, $\text{SHARE}_{\text{P}}^{\text{TH}}$ 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: In Step 1, the dealer distributes a bivariate polynomial $g(x, y)$, that contains a d-sharing of its input s. It follows from the properties of a bivariate polynomial that $g(x, y)$ reveals no more information about s than the d-sharing. After Step 1, the adversary does not obtain any additional information. 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, the shares held by correct parties are always consistent, which implies already 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 always outputs some correct d-sharing. □

Protocol $\text{SHARE}_{\text{P}}^{\text{TH}}$: Given input s from the dealer, compute a d-sharing $[s]$ of this value.

1. The dealer chooses a random (2-dimensional) polynomial $g(x, y)$ with $g(0, 0) = s$, of degree d in both variables, and sends to party p_i (for $i = 1, \ldots, n$) the (1-dimensional) 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), party p_i sends $h_i(\alpha_j)$ to party p_j, and party p_j checks whether $h_i(\alpha_j) = k_j(\alpha_i)$. If this check fails, it broadcasts a complaint, and the dealer has to broadcast the correct value.

3. If some party p_i observes an inconsistency between the polynomials received in Step 1 and the broadcasted value in Step 2, it accuses the dealer. The dealer has to answer 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.

4. If the dealer does not answer some complaint or accusation, or if the broadcasted values contradict, the parties output a default d-sharing. Otherwise, each party p_i outputs $s_i := k_i(0)$, and the dealer outputs $\hat{s}(x) := g(x, 0)$.[8]

Figure 4.5: The perfectly secure share protocol for threshold adversaries.

The public reconstruction[9] of a d-shared value s uses techniques from coding theory, which allow a more intuitive understanding of the trade-off between correctness and robustness. It follows from coding theory that a d-sharing is equivalent to a code based on the evaluation of a polynomial of degree d. Such a code has minimal distance $n - d$. Hence, the decoding algorithm can detect up to $n - d - 1$ errors and abort (for correctness), or correct up to $\lfloor \frac{n-d-1}{2} \rfloor$ errors (for robustness). In our protocol, we trade correctness for

[8] That means, in general we discard the second dimension of $g(x, y)$. Yet, in a special context, we will subsequently make use of it.

[9] Private reconstruction can be reduced to public reconstruction using a blinding technique as discussed in Section 2.3.3. Note that the trivial solution, where each party sends its share to party p_k, does not achieve agreement on abort.

Protocol PUBLIC RECONSTRUCTION$_{\mathrm{P}}^{\mathrm{TH}}$: Given a d-sharing $[s]$ of some value s, reconstruct s to all parties.

1. Each party p_i broadcasts its share s_i. Let $\vec{s} = (s_1, ..., s_n)$ denote the vector of broadcasted shares.
2. Each party identifies the closest codeword $\vec{s}_c$ (e.g. using the Berlekamp-Welch algorithm). If the Hamming distance between $\vec{s}_c$ and $\vec{s}$ is larger than e, the protocol is aborted. Otherwise, each party interpolates the entries in $\vec{s}_c$ with a polynomial $\hat{s}_c(x)$ of degree d, and outputs $\hat{s}_c(0)$.[10]

Figure 4.6: The perfectly secure public reconstruction protocol for threshold adversaries.

robustness by introducing the correction parameter $e < \frac{n-d}{2}$: Our decoding algorithm provides error correction for up to e errors, and error detection for up to $(n-d)-e-1$ errors. Note that this trade-off is optimal: If the distance to the correct codeword is greater than $(n-d)-e-1$, the distance to the next codeword is at most e, and the decoding algorithm would decode to the wrong codeword.

The public reconstruction protocol (Figure 4.6) proceeds as follows: First, each party broadcasts its share s_i. Then, each party locally "decodes" the broadcasted shares to the closest codeword, and aborts if the number of errors cannot be explained with an adversary actively corrupting at most e parties, i.e., if the Hamming distance between the shares and the decoded codeword is larger than e. PUBLIC RECONSTRUCTION$_{\mathrm{P}}^{\mathrm{TH}}$ is the only subprotocol that might abort. All other protocols abort only if they use PUBLIC RECONSTRUCTION$_{\mathrm{P}}^{\mathrm{TH}}$ as a subprotocol and the invocation thereof aborts. Therefore, it is sufficient to discuss agreement on abort only for this protocol.

Lemma 11. *Let d be the sharing degree, and e be the robustness parameter, where $d + 2e < n$. Given a d-sharing $[s]$ of some value s,* PUBLIC RECONSTRUCTION$_{\mathrm{P}}^{\mathrm{TH}}$ *is correct if $|\mathcal{D}^*| < (n-d)-e$, is robust if $|\mathcal{D}^*| \leq e$, and always guarantees agreement on abort.*

[10] That means, in general we discard the vector $\vec{s}_c$ of corrected shares. Yet, in a special context, we will subsequently make use of it.

Proof. Only actively corrupted parties broadcast incorrect shares. Hence, the Hamming distance between the broadcasted shares and the correct codeword is at most $|\mathcal{D}^*|$.

CORRECTNESS: The minimal distance between two codewords is $(n-d)$, and the decoding algorithm corrects up to e errors. Hence, if $|\mathcal{D}^*| + e < (n-d)$, the decoding algorithm never decodes to the incorrect codeword.

ROBUSTNESS: Since $|\mathcal{D}^*| \leq e$, the Hamming distance between the shares and the correct codeword is at most e and the decoding cannot be aborted.

Since the abort decision is based only on broadcasted values, we always have AGREEMENT ON ABORT (cf. Section 2.3.2). □

4.2.1.2 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 $[r] = [r_1] + \ldots + [r_n]$.

The multiplication protocol (Figure 4.7) is more involved. The product c of two 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 $\hat{v}(0)$. This results in a d-sharing of the product c.

This protocol is secure only against passive adversaries. An active adversary could share a wrong value $v_i' \neq v_i$. Therefore, each party has to prove that it shared the correct value $v_i = a_i b_i$ by invoking ABC-PROOF$_{\text{P}}^{\text{TH}}$ on d-sharings of v_i, a_i, and b_i. To obtain the d-sharings of a_i and b_i, we invoke UPGRADE$_{\text{P}}^{\text{TH}}$ on the d-sharings of a and b, resulting in $[a_i]$ and $[b_i]$ for all i.

Given $[a_i]$, $[b_i]$, and $[v_i]$, it remains to show that $a_i b_i = v_i$, which is equivalent to $z = 0$ for $[z]^{2d} := [a_i][b_i] - [v_i]$, where $[z]^{2d}$ is a $2d$-sharing. Party p_i knows the sharing polynomial $\hat{g}(x)$ corresponding to $[z]^{2d}$. However, party p_i cannot simply broadcast $\hat{g}(x)$, since this would violate secrecy (the adversary could obtain information about other shares). Therefore, we blind $[z]^{2d}$ by adding a uniformly random $2d$-sharing of 0.

Finally, all parties (locally) check whether $z = 0$, and whether party p_i broadcasted the correct polynomial $\hat{g}(x)$, i.e. for party p_j whether

$\hat{g}(\alpha_j) = [z]_j^{2d}$. Two polynomials of degree $2d$ are equal if they coincide in $2d+1$ points. So, if party p_i broadcasts an incorrect $\hat{g}(x)$, and if there are at least $2d+1$ correct parties, at least one correct party detects the cheating attempt and raises an accusation. To prove the accusation, the shares of the corresponding party are reconstructed.

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

1. Each party p_i computes $v_i = a_i b_i$, and invokes SHARE$_{\mathrm{P}}^{\mathrm{TH}}$ on v_i, resulting in $[v_i]$.
2. Invoke UPGRADE$_{\mathrm{P}}^{\mathrm{TH}}$ on $[a]$ and $[b]$, resulting in $[a_i]$ and $[b_i]$ for $i = 1, \ldots, n$.
3. For $i = 1, \ldots, n$, all parties invoke ABC-PROOF$_{\mathrm{P}}^{\mathrm{TH}}$ on $[a_i]$, $[b_i]$, and $[v_i]$. If the proof is rejected, invoke PUBLIC RECONSTRUCTION$_{\mathrm{P}}^{\mathrm{TH}}$ on $[a_i]$ and $[b_i]$, and use a default d-sharing of $v_i := a_i b_i$.
4. All parties distributedly compute the Lagrange interpolation[11] on $[v_1], \ldots, [v_n]$ for $c = v(0)$, and output the resulting $[c]$.

Figure 4.7: The perfectly secure multiplication protocol for threshold adversaries.

Lemma 12. *Let d be the sharing degree, and e be the robustness parameter, where $d + 2e < n$. Given d-sharings of a and b,* SHARE$_{\mathrm{P}}^{\mathrm{TH}}$ *outputs a correct d-sharing of the product $c = ab$ if $2d < n$ and if the subprotocols are correct, is secret if the subprotocols are secret and correct, and robust if the subprotocols are robust.*

Proof. By assumption, all subprotocols are secure. In Step 4, the parties interpolate a polynomial of degree $2d$ using n evaluation points. This interpolation computes the correct result only if $2d < n$, which is given by assumption. Hence, security of the multiplication protocol follows straightforwardly. □

We first present the UPGRADE$_{\mathrm{P}}^{\mathrm{TH}}$ protocol (Figure 4.8): Given a d-sharing $[s]$ for some value s, the UPGRADE$_{\mathrm{P}}^{\mathrm{TH}}$ protocol computes d-sharings $[s_i]$ of all shares s_i.

[11] Lagrange interpolation is a linear and therefore local computation on the shares of $v_1, \ldots, v_n$.

Protocol UPGRADE$_{\mathrm{P}}^{\mathrm{TH}}$: Given $[s]$, compute $[s_i]$ for $i = 1, \ldots, n$.

1. All parties jointly compute a sharing of a random value r, such that each share r_j is also shared with a d-sharing:
   (a) Each party p_i chooses a uniformly random value $r^{(i)}$, and invokes SHARE$_{\mathrm{P}}^{\mathrm{TH}}$ on $r^{(i)}$. By keeping the second dimension at the end of SHARE$_{\mathrm{P}}^{\mathrm{TH}}$, this results in a d-sharing $[r^{(i)}]$, where additionally every share $r_j^{(i)}$ is d-shared with $[r_j^{(i)}]$.
   (b) All parties compute $[r] = \sum_{i=1}^{n} [r^{(i)}]$ and $[r_j] = \sum_{i=1}^{n} [r_j^{(i)}]$ for $j = 1, \ldots, n$.
2. All parties compute $[q] := [r] - [s]$ and invoke PUBLIC RECONSTRUCTION$_{\mathrm{P}}^{\mathrm{TH}}$ on $[q]$. Denote by $q_1, \ldots, q_n$ the (error-corrected) shares, which are known to all parties.
3. For $j = 1 \ldots n$, all parties compute $[s'_j] = [r_j] - q_j$, where s'_j is a share of some value s'.
4. Each party p_i outputs its share s'_i of s', a d-sharing $[s'_i]$ of s'_i, and for all j a share-share of s'_j.

Figure 4.8: A perfectly secure protocol for threshold adversaries for upgrading a d-sharing.

Lemma 13. *Let d be the sharing degree, and e be the robustness parameter, where $d + 2e < n$, and assume that* SHARE$_{\mathrm{P}}^{\mathrm{TH}}$ *and* PUBLIC RECONSTRUCTION$_{\mathrm{P}}^{\mathrm{TH}}$ *are secure. Given a d-sharing of some value s,* UPGRADE$_{\mathrm{P}}^{\mathrm{TH}}$ *correctly, robustly, and secretly computes a d-sharing of each share s_i.*

Proof. The proof follows directly from the observation that the protocol is as secure as SHARE$_{\mathrm{P}}^{\mathrm{TH}}$ and PUBLIC RECONSTRUCTION$_{\mathrm{P}}^{\mathrm{TH}}$: Let $\hat{r}(x, y)$ be the polynomial with which r is shared, and $\hat{s}(x)$ be the polynomial with which s is shared.

CORRECTNESS follows from the observation that in Step 3 the parties compute $\hat{h}(x, y) = \hat{r}(x, y) - \hat{q}(x) = \hat{r}(x, y) - (\hat{r}(x, 0) - \hat{s}(x))$. Hence, $\hat{h}(x, 0) = \hat{s}(x)$ (and $s'_i = s_i$ and $s' = s$) and $h(\alpha_i, y)$, the polynomial corresponding to $[s_i]$, is a random[12] polynomial of degree d, as required.

[12] Actually, the d-sharings $[s_i]$ are only $(d+1)$-wise independent. However, this does not affect

SECRECY is guaranteed in the sense that the protocol does not leak more information than the specified output.

ROBUSTNESS: This security requirement depends only on the subprotocols. □

Next, we present a protocol that allows to prove that a given sharing contains the product of the values of two other given sharings (Figure 4.9).

Lemma 14. *Let d be the sharing degree, and e be the robustness parameter, where $d + 2e < n$, and assume that* $\text{SHARE}_{\text{P}}^{\text{TH}}$ *and* PUBLIC RECONSTRUCTION$_{\text{P}}^{\text{TH}}$ *are secure. Given d-sharings $[a_i]$, $[b_i]$, and $[v_i]$,* ABC-PROOF$_{\text{P}}^{\text{TH}}$ *is correct if $|\mathcal{D}^*| < n - 2d$, is secret given that the subprotocols are correct, and is always robust.*

Proof. SECRECY: If party p_i is correct, then $[0]^{2d}$ (Step 1) is a uniformly random $2d$-sharing of the value 0 [BGW88]: Let $\hat{r}_i(x)$ be the sharing polynomial (of degree d and with uniformly random coefficients) corresponding to $[r_i]$. Then, the sharing polynomial corresponding to $[0]^{2d}$ is $x^d\hat{r}_d(x)+\ldots+x^1\hat{r}_1(x)$. It is easy to see that this is a polynomial of degree $2d$ with uniformly random coefficients. This implies that the sharing $[z]^{2d}$ computed in Step 2 is also a uniformly random $2d$-sharing. If in Step 5 the shares of a party p_j are reconstructed, either party p_i or party p_j are actively corrupted. Hence, the adversary knew these shares already beforehand.

CORRECTNESS: We are guaranteed that the sharing degree of $[a_i]$, $[b_i]$, and $[v_i]$ is d (by assumption on the input), that $[0]^{2d}$ is a $2d$-sharing containing the value 0 (by construction and correctness of $\text{SHARE}_{\text{P}}^{\text{TH}}$), and that $\hat{g}(x)$ is a polynomial of degree $2d$ (by construction). Now, if $\hat{g}(x) = 0$ and $\hat{g}(x)$ is the sharing polynomial corresponding to $[z]^{2d} = [0]^{2d} + [a_i][b_i] - [v_i]$, then $v_i = a_ib_i$. Therefore, we only need to verify that the broadcasted polynomial $\hat{g}(x)$ is correct.

In Step 3, the sharing polynomial of $[z]^{2d}$ is fixed. Assume that, in this step, party p_i broadcasts a wrong polynomial $\hat{g}'(x)$.[13] Note that $\hat{g}'(x)$ has to be also of degree $2d$. Now, if two polynomials of degree $2d$ coincide in at least $2d + 1$ points, then they are equal. Hence, if $n - |\mathcal{D}^*| \geq 2d + 1$ (i.e. if there are at least $2d + 1$ correct parties), and if for each correct party p_j it holds that $\hat{g}(\alpha_j) = [z]_j^{2d}$, the polynomials must be equal.

---

security.

[13] E.g. one that hides the fact that $v_i \neq ab$, i.e. the adversary sets $\hat{g}'(0) = 0$ and selects $2d$ evaluation points α where $\hat{g}'(\alpha) = \hat{g}(\alpha)$.

Protocol ABC-PROOF$_{\mathrm{P}}^{\mathrm{TH}}$: Given $[a_i]$, $[b_i]$, and $[v_i]$ that are known to party p_i, check whether $v_i = a_i b_i$.

1. All parties compute a $2d$-sharing of 0, such that party p_i knows the sharing polynomial [BGW88]:
   (a) Party p_i chooses uniformly random values $r_1, \ldots, r_d$ and invokes SHARE$_{\mathrm{P}}^{\mathrm{TH}}$ with degree d on each of the values, resulting in $[r_1], \ldots, [r_d]$.
   (b) All parties compute $[0]^{2d} = x^d[r_d] + \ldots + x^1[r_1]$.
   (i.e. each party p_j computes $[0]_j^{2d} = \alpha_j^d[r_d]_j + \ldots + \alpha_j^1[r_1]_j$)
2. (a) All parties compute $[w]^{2d} := [a_i][b_i]$, i.e. each party p_j locally computes $[w]_j^{2d} := [a_i]_j[b_i]_j$.
   (b) All parties compute $[z]^{2d} := [0]^{2d} + [w]^{2d} - [v_i]$, i.e. each party p_j locally computes$[z]_j^{2d} := [0]_j^{2d} + [w]_j^{2d} - [v_i]_j$.
   (Note that party p_i knows the sharing polynomial $\hat{g}(x)$ of $[z]^{2d}$.)
3. Party p_i broadcasts $\hat{g}(x)$. If $\hat{g}(x)$ has a degree greater than $2d$ or $\hat{g}(0) \neq 0$, all parties output *reject*.
4. Each party p_j checks that $\hat{g}(\alpha_j) = [z]_j^{2d}$. If this check fails, it raises a complaint.
5. For each complaining party p_j, all parties open the four shares $[a_i]_j$, $[b_i]_j$, $[v_i]_j$, and $[0]_j^{2d}$. The complaint holds if
   $$\hat{g}(\alpha_j) \neq [0]_j^{2d} + [a_i]_j[b_i]_j - [v_i]_j.$$
   For the opening of $[a_i]_j$, invoke UPGRADE$_{\mathrm{P}}^{\mathrm{TH}}$ on $[a_i]$, and then PUBLIC RECONSTRUCTION$_{\mathrm{P}}^{\mathrm{TH}}$ on the d-sharing of the share $[a_i]_j$ ($[b_i]_j$ and $[v_i]_j$ accordingly). Opening of $[0]_j^{2d}$ is done by opening $[r_1]_j, \ldots, [r_d]_j$, and computing $[0]_j^{2d} = \alpha_j^d[r_d]_j + \ldots + \alpha_j^1[r_1]_j$.
6. If any complaint holds, output *reject*. Otherwise output *correct*.

Figure 4.9: A perfectly secure protocol for threshold adversaries for proving that $c = ab$.

Furthermore, correctness can be violated if the adversary can provoke a

correct proof to be rejected. For this purpose, the adversary has to provoke an incorrect opening in Step 5, which is excluded by assumption.

ROBUSTNESS: This security requirement depends only on the subprotocols. □

4.2.1.3 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 $\pi_{\mathrm{P}}^{d,e}$:

Lemma 15. *Let d be the sharing degree, and e be the robustness parameter, where $d + 2e < n$. Protocol $\pi_{\mathrm{P}}^{d,e}$ guarantees*

- CORRECTNESS *if $|\mathcal{D}^*| < (n - d) - e$ and $|\mathcal{D}^*| < n - 2d$,*
- SECRECY *if $|\mathcal{E}^*| \leq d$ and correctness is guaranteed,*
- ROBUSTNESS *if $|\mathcal{D}^*| \leq e$, and*
- AGREEMENT ON ABORT *always.*

Proof. For each property, given its condition in the lemma, it is easy to verify that the corresponding conditions for this property in all subprotocols are fulfilled. □

4.2.2 The Main Result for Threshold Adversaries

The following theorem states the optimal bound for perfectly secure reactive MPC with graceful degradation of both security (allowing for hybrid security) and corruptions (allowing for mixed adversaries) for threshold adversaries, given broadcast. The model without broadcast is treated in Section 4.3. Furthermore, we show that the bound is sufficient for MPC by providing parameters for the generalized protocol introduced in Section 4.2.1.

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

$$\left(t_a^c + t_p^s + t_a^r < n \ \wedge \ t_a^c + 2t_p^s < n\right) \quad \vee \quad t_p^s = 0.$$

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

Note that the bound holds for any $(t_a^f, t_p^f) \leq (t_a^s, t_p^s)$, and we can always have $(t_a^f, t_p^f) = (t_a^s, t_p^s)$.

Proof. Necessity of the bound in the theorem follows directly from the corresponding proof for general adversaries (Section 4.1.4). To prove that the bound is also sufficient, we first note that if $t_p^s = 0$, there is no secrecy requirement, and we can directly use the trivial non-secret protocol described in Appendix 4.1.2. Otherwise, we employ the parametrized version $\pi_{\mathrm{P}}^{d,e}$ of the protocol of [BGW88] described in Section 4.2.1 with $d := t_p^s$ and $e := \max(t_a^r, t_a^f)$.

The precondition of Lemma 15 is that $d + 2e < n$. Note that $e = \max(t_a^r, t_a^f) \leq t_a^c$. Hence for our choice of parameters d and e, we have that $d + e + e \leq t_p^s + t_a^c + \max(t_a^r, t_a^f)$. If $\max(t_a^r, t_a^f) = t_a^r$, the precondition of the lemma follows from the left-hand side of the condition in the theorem. Otherwise, it follows from the right-hand side (note that $t_a^f \leq t_p^s$). Hence, we can apply the lemma 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.

For fairness, analogously to the setting with general adversaries, we have $t_a^f \leq e$. Hence, for $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq (t_a^f, t_p^f)$ the protocol is robust, and the adversary cannot abort. □

4.3 The Model without Broadcast Channel

We now turn to a model where no broadcast channels are given, but only a complete network of secure channels. In this model, we obtain essentially the same bound as in the model with broadcast channels, with an additional limitation on the sets of possibly actively corrupted parties. If this additional condition is not fulfilled, no security guarantees (not even agreement on abort) are given. For the bounds, we assume that correctness requires agreement on abort.

Corollary 1. *In the secure channels model without broadcast and with general adversaries, perfectly secure reactive MPC among $n \geq 2$ parties with respect to $(\mathcal{Z}^c, \mathcal{Z}^s, \mathcal{Z}^r, \mathcal{Z}^f)$, where $\mathcal{Z}^r \subseteq \mathcal{Z}^f \subseteq \mathcal{Z}^c$ and $\mathcal{Z}^s \subseteq \mathcal{Z}^c$, is possible if*

$$\Big(\big(\forall(\mathcal{D}^c, \cdot) \in \mathcal{Z}^c, (\cdot, \mathcal{E}_1^s), (\cdot, \mathcal{E}_2^s) \in \mathcal{Z}^s, (\mathcal{D}^f, \cdot) \in \mathcal{Z}^f : \\ \mathcal{D}^c \cup \mathcal{E}_1^s \cup \mathcal{D}^f \neq \mathcal{P} \wedge \mathcal{D}^c \cup \mathcal{E}_1^s \cup \mathcal{E}_2^s \neq \mathcal{P}\big) \ \vee \ \mathcal{Z}^s = \{(\emptyset, \emptyset)\}\Big)$$

$$\wedge \quad \forall(\mathcal{D}_1^c, \cdot), (\mathcal{D}_2^c, \cdot), (\mathcal{D}_3^c, \cdot) \in \mathcal{Z}^c : \mathcal{D}_1^c \cup \mathcal{D}_2^c \cup \mathcal{D}_3^c \neq \mathcal{P}.$$

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

Proof. The sufficiency of the bound follows directly by basing the protocol of Section 4.1.1 on the perfectly secure broadcast protocol of [FM98]. Given the condition in the corollary, the broadcast protocol is secure. Otherwise, no security guarantees (not even agreement on abort) are given.

The necessity of the bound follows from the necessity of Theorem 1 and from [LSP82, Lam83]: As broadcast is a special type of MPC, a general MPC protocol must be able to achieve broadcast. Yet, even non-robust but correct broadcast (or weak Byzantine agreement as it is more commonly called in the literature) cannot be realized with perfect security if three sets of possibly actively corrupted parties may cover the entire player set [Lam83]. Thus, perfectly secure MPC without broadcast requires $\forall(\mathcal{D}_1^c,\cdot),(\mathcal{D}_2^c,\cdot),(\mathcal{D}_3^c,\cdot) \in \mathcal{Z}^c : \mathcal{D}_1^c \cup \mathcal{D}_2^c \cup \mathcal{D}_3^c \neq \mathcal{P}$, in addition to the bound of Theorem 1. □

An analogous statement holds for the setting with threshold adversaries.

Corollary 2. *In the secure channels model without broadcast and with threshold adversaries, perfectly secure reactive MPC among $n \geq 2$ parties with thresholds (t_a^c, t_p^c), (t_a^s, t_p^s), (t_a^r, t_p^r), and (t_a^f, t_p^f), where $(t_a^s, t_p^s) \leq (t_a^c, t_p^c)$ and $(t_a^r, t_p^r) \leq (t_a^f, t_p^f) \leq (t_a^c, t_p^c)$, is possible if*

$$\Big(\big(t_a^c + t_p^s + t_a^f < n \ \wedge \ t_a^c + 2t_p^s < n\big) \quad \vee \quad t_p^s = 0\Big) \quad \wedge \quad 3t_a^c < n$$

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

Proof. The sufficiency of the bound follows directly by basing the protocol of Section 4.2.1 on the perfectly secure broadcast protocol of [BGP89, CW89] tolerating $|\mathcal{D}^*| < \frac{n}{3}$ actively corrupted parties. Again, for $|\mathcal{D}^*| \geq \frac{n}{3}$, no security guarantees (not even agreement on abort) are given. The necessity of the bound follows along the lines of the setting with general adversaries. □

4.4 Summary

We have provided perfectly secure MPC protocols with graceful degradation of both security and corruptions, for both threshold and general adver-

saries. These protocols are strict generalizations (and combinations) of protocols with hybrid security and for mixed adversaries. Furthermore, we derived tight bounds for the existence of such protocols, and have proven that our protocols achieve these bounds.

A different perspective on our results reveals that our protocols provide an additional degree of freedom: Ben-Or et al. [BGW88] prove that in a setting with full security, the condition $3t < n$ is optimal, where t denotes the number of actively corrupted parties. This condition leaves a single optimal choice for t. Fitzi et al. [FHM98] consider mixed adversaries, and thus generalize the original condition to $2t_p + t_a < n$. This condition allows to choose a threshold t_p, thereby fixing an optimal value for t_a. The bound in Theorem 2 illustrates that our work further generalizes the known results: Our treatment of graceful degradation provides an additional degree of freedom, allowing to choose thresholds t_p^s for secrecy and t_a^r for robustness, thereby fixing an optimal threshold t_a^c for correctness. A similar observation holds for our main result for general adversaries (Theorem 1). As a consequence, we provided the most comprehensive treatment, integrating the previous results on perfect security.

Furthermore, our protocols provide a flexible tradeoff between different security properties. Consider a setting with $n = 100$ parties. The traditional results from [BGW88] provide one protocol, which is secure for up to 49 passive corruptions, and another protocol for up to 33 active corruptions. In [FHM98], these protocols are merged into a single protocol which is secure e.g. for 40 passively corrupted parties, of which 19 may even be actively corrupted. Yet, if only one more party is corrupted (passively or actively), immediately, all security guarantees are lost. Our protocols allow to trade one security guaranteed (which, in a certain situation, might be less important) for another security guarantee (which is more important). For example, we provide a protocol that is at the same time

- secret for up to 12 passive corruptions,
- robust for up to 12 active corruptions (and any number of passive corruptions), and
- correct for up to 75 active corruptions (and any number of passive corruptions).

If more robustness is required, the threshold for correctness can be reduced e.g. to 50 active corruptions. This results in a protocol that guarantees robustness for up to 37 active corruptions.

Finally, our results confirm the folklore belief that, in the perfect setting with mixed adversaries, passive corruption affects only secrecy, but not correctness or robustness.

Chapter 5

Protocols with Statistical Security

For the setting with statistical security, we use the same approach as in the perfect setting: First, we design parametrized protocols and analyze them with respect to correctness, secrecy, and robustness (again, fairness is treated separately). Then, given the security of the parametrized protocols, we deduce tight bounds for reactive MPC.

Our model with graceful degradation of both security and corruptions allows to study the exact consequences of active and passive corruptions on the various security properties. In the perfect setting, we have seen that the number of passive corruptions only 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 inevitably (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. 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. Here, we show that this assumption is too strong, and that signatures can be used even without an honest majority. In Sections 5.2 and 5.3, we provide optimal protocols for both general and threshold adversaries, respectively, that cope with this issue.

The treatment follows along the lines of [HLMR12], where this setting was first discussed.

5.1 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 choose and send α_{ij} to p_j before the protocol starts.

Definition 4 (α-consistent)**.** A triple (v, y, z) is α-consistent, if the points $(0, v)$, $(1, y)$, and (α, z) lie on a line, i.e. if $z = (y - v)\alpha + v$.

Definition 5 (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 (correct) $p_k \in \mathcal{P}$ 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 $\langle v \rangle_{i,j}$.

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 $v + v'$.

5.1.1 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, z_k and z'_k uniformly at random 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.

Figure 5.1: The statistically secure IC-Sign protocol.

Lemma 16. *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 $\perp$.* 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. □

5.1.2 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.

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.

Figure 5.2: The statistically secure IC-Reveal protocol.

Lemma 17. *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:*

1. *If p_j is correct, all correct parties p_k output $x_k = (\text{"accept"}, v)$.*
2. *Else, if both p_i and p_k are honest, then $x_k \in \{(\text{"accept"}, v), \text{"reject"}\}$ (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 $x_k = (\text{"accept"}, v')$ 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. □

5.2 MPC with General Adversaries

Our protocol for general adversaries is based on [HMZ08], which is an adaptation of the perfectly secure protocol of [Mau02, BFH+08] to the statistical case. For a generic protocol construction, it is sufficient to consider two parameters (cf. Section 4.1): 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 robustness parameter $\mathcal{R}$. In contrast to the perfect case, here we need to consider both active and passive corruption. Therefore, the robustness 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.

5.2.1 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. Fairness is discussed in Section 5.2.2. 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.

5.2.1.1 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 Section 5.1.

Definition 6 (Group Commitment)**.** A group $\mathcal{G}$ is *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.

Figure 5.3: The statistically secure Group Commit protocol for a group $\mathcal{G}$ for general adversaries.

Lemma 18. *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 while providing secrecy with respect to the remaining parties $\mathcal{P} \setminus \mathcal{G}$. 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 the intermediary

are actively corrupted. Second, if all parties in $\mathcal{G}$ are correct (i.e., in all invocations of IC-SIGN, both the signer and the intermediary 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).

Lemma 19. *Given the robustness 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 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 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.

First, we 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

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 $\langle v \rangle_{j,i}$ (i.e., p_i opens all signatures it holds from parties $p_j \in \mathcal{G}$ on v).
   (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) or 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.

Figure 5.4: The statistically secure Group Reveal protocol for a group $\mathcal{G}$ for general adversaries.

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 $\mathcal{G} \setminus \mathcal{D} \notin \mathcal{D}^*$**:** 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 $\mathcal{G} \setminus \mathcal{D} \notin \mathcal{D}^*$.

Case $\mathcal{G} \notin \mathcal{E} \ \wedge \ \mathcal{P} \setminus \mathcal{E} \notin \mathcal{D}^*$**:** Since $\mathcal{G} \notin \mathcal{E}$, 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 $\mathcal{P} \setminus \mathcal{E} \notin \mathcal{D}^*$.

Case $\mathcal{G} \notin \mathcal{E}^* \ \wedge \ \mathcal{P} \setminus \mathcal{E}^* \notin \mathcal{D}$**:** 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.

Since the abort decision is based only on broadcasted values, we always have AGREEMENT ON ABORT (cf. Section 2.3.2). □

5.2.1.2 The Underlying Verifiable Secret Sharing Scheme

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 group-committed to it.

Definition 7 ($\mathcal{S}$-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 *$\mathcal{E}$-secret* if the summands held by the parties in $\mathcal{E}$ are statistically independent from the secret, and *$\mathcal{D}$-permissive* if the summands held by the parties in $\mathcal{P} \setminus \mathcal{D}$ uniquely define the secret.

Lemma 20. *An $\mathcal{S}$-sharing is $\mathcal{E}$-secret if $\exists S_i \in \mathcal{S} : S_i \cap \mathcal{E} = \emptyset$, and $\mathcal{D}$-permissive if $\forall S_i \in \mathcal{S} : S_i \setminus \mathcal{D} \neq \emptyset$.*

Proof. Secrecy follows from the fact that $\mathcal{E}$ lacks at least one summand s_i, and from the secrecy of group commitments. Furthermore, given that $\forall S_i \in \mathcal{S} : S_i \setminus \mathcal{D} \neq \emptyset$, each summand s_i is held by at least one party in $\mathcal{P} \setminus \mathcal{D}$. 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 Figure 5.5).

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

1. The dealer chooses uniformly random summands $s_1, \ldots, s_\ell$ such that $s = \sum_{i=1}^{\ell} s_i$, 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.

Figure 5.5: The statistically secure share protocol for general adversaries.

Lemma 21. *Let $\mathcal{S}$ be the sharing specification. On input s from the dealer, $\text{SHARE}_{\mathcal{S}}^{\text{GA}}$ 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[14] of a shared value (Figure 5.6), 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$_{\mathcal{S}}^{\mathbf{GA}}$: 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 $s = s_1 + \ldots + s_\ell$.

Figure 5.6: The statistically secure public reconstruction protocol for general adversaries.

Lemma 22. *Given the sharing specification $\mathcal{S}$, the robustness parameter $\mathcal{R}$, and an $\mathcal{S}$-sharing of some value s,* PUBLIC RECONSTRUCTION$_{\mathcal{S}}^{\mathrm{GA}}$ *reconstructs s to all parties. The protocol is correct if*

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

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

[14] Private reconstruction can be reduced to public reconstruction using a blinding technique as discussed in Section 2.3.3.

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. □

5.2.1.3 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 $r = r_1 + \ldots + r_n$ (see Section 2.3.3).

For the multiplication of two values a and b, we adapt the protocol from [HMZ08] by using our modified share and reconstruct protocols (Figure 5.7). 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. In order to prove that the sharing contains the correct value, the corresponding party provides a zero-knowledge proof by invoking ABC-PROOF$_{\mathrm{S}}^{\mathrm{GA}}$ 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$_{\mathrm{S}}^{\mathrm{GA}}$, a subprotocol that allows to share individual summands.

Lemma 23. *Given the sharing specification $\mathcal{S}$, the robustness parameter $\mathcal{R}$, and $\mathcal{S}$-sharings of a and b,* MULTIPLICATION$_{\mathrm{S}}^{\mathrm{GA}}$ *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. □

Protocol MULTIPLICATION$_{\mathcal{S}}^{\mathrm{GA}}$: 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$_{\mathrm{S}}^{\mathrm{GA}}$ on it, resulting in $[v_{ij}]$.
   (b) Invoke GROUPSHARE$_{\mathrm{S}}^{\mathrm{GA}}$ 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$_{\mathrm{S}}^{\mathrm{GA}}$ on $[a_i]$, $[b_j]$, and $[v_{ij}]$ with prover p_{ij}. If the proof is rejected, invoke PUBLIC RECONSTRUCTION$_{\mathrm{S}}^{\mathrm{GA}}$ 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 $c = ab$.

Figure 5.7: The statistically secure multiplication protocol for general adversaries.

The GROUPSHARE$_{\mathrm{S}}^{\mathrm{GA}}$ subprotocol (Figure 5.8) 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$_{\mathrm{S}}^{\mathrm{GA}}$ computes an $\mathcal{S}$-sharing $[s_i]$ of the value s_i, given that at least one party in S_i is correct.

Lemma 24. *Given the sharing specification $\mathcal{S}$, the robustness 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$_{\mathrm{S}}^{\mathrm{GA}}$ *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

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

1. p invokes SHARE$_S^{GA}$ on s_i, resulting in $[s_i]$.
2. For each $p_j \in S_i$: Invoke PRIVATE RECONSTRUCTION$_S^{GA}$ 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 $[s_i]$.

Figure 5.8: The statistically secure group share protocol for general adversaries.

$[a]$ and $[b]$ (Figure 5.9). The protocol is along the lines of the protocol in [CDD+99].

Lemma 25. *Given are the sharing specification $\mathcal{S}$, the robustness 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$_S^{GA}$ *outputs "accept". If $c \neq ab$, then* ABC-PROOF$_S^{GA}$ *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 $z = 0$.

Protocol ABC-PROOF$_S^{GA}$: 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$_S^{GA}$ 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$_S^{GA}$ on $[b'']$.
   (d) Compute $[z] = b''[a] - r[c] - [c']$ and invoke PUBLIC RECONSTRUCTION$_S^{GA}$ 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".

Figure 5.9: The statistically secure protocol for proving that $c = ab$ for general adversaries.

ROBUSTNESS: It follows from inspection that the protocol does not abort. □

5.2.1.4 The Security of the Generalized Protocol from [HMZ08]

Considering the security of the subprotocols described above, we can derive the security of the parametrized protocol, denoted by $\pi_S^{\mathcal{S},\mathcal{R}}$:

Lemma 26. *Given the sharing specification $\mathcal{S}$ and the robustness parameter $\mathcal{R}$, the protocol $\pi_S^{\mathcal{S},\mathcal{R}}$ guarantees* correctness *if*

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

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

Proof. $\pi_S^{\mathcal{S},\mathcal{R}}$ provides a certain security guarantee against $(\mathcal{D}^*, \mathcal{E}^*)$ if all subprotocols (cf. Lemmas 18, 19, and 21 to 25) and the sharing (cf. Lemma 20)

provide this guarantee against $(\mathcal{D}^*, \mathcal{E}^*)$. For each guarantee, it can easily be verified that the condition in the lemma implies the corresponding conditions in the mentioned lemmas. □

5.2.2 Main Result

The following theorem states the optimal bound for statistically secure reactive 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 following section, we prove that the bound is also necessary.

Theorem 3. *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*

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

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

Proof. The necessity of the bound in the theorem is proven in Section 5.2.3. To prove that the bound is also sufficient, we first note that if $\mathcal{Z}^s = \{(\emptyset, \emptyset)\}$, there is no secrecy requirement, and we can directly use the trivial non-secret protocol described in Section 4.1.2. Otherwise, we employ the protocol $\pi_{\mathrm{S}}^{\mathcal{S},\mathcal{R}}$ described in Section 5.2.1. 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 26 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.

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. □

5.2.3 Proofs of Necessity

In this section, we prove that the bound in Theorem 3 is necessary, i.e. if violated, MPC is impossible.[15] The bound is violated if

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

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].

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 $\mathcal{Z}^s \neq \{(\emptyset,\emptyset)\} \wedge$
$\exists(\cdot,\mathcal{E}^s) \in \mathcal{Z}^s, (\mathcal{D}^r,\mathcal{E}^r) \in \mathcal{Z}^r, (\mathcal{D}^c,\mathcal{E}^c) \in \mathcal{Z}^c :$
$\mathcal{D}^c \cup \mathcal{D}^r \cup \mathcal{E}^s = \mathcal{P} \quad \wedge \quad (\mathcal{E}^s \cup \mathcal{E}^r = \mathcal{P} \ \vee\ \mathcal{D}^c \cup \mathcal{E}^r = \mathcal{P})$
$\wedge \quad (\mathcal{E}^s \cup \mathcal{E}^c = \mathcal{P} \ \vee\ \mathcal{D}^r \cup \mathcal{E}^c = \mathcal{P})$.
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

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

$\mathcal{E}^c = \mathcal{D}^c \cup \mathcal{E}^s$. 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 27: 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 along the lines of the proof in Section 4.1.4.1 for the setting with perfect security. In contrast to the perfect setting, here we also need to consider the state generation. We extend the corresponding definition accordingly.

Definition 8 (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 27. *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.

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 due to completeness), 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 due to completeness), 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. □

5.3 MPC with Threshold Adversaries

In the perfect setting (Section 4.2), it was sufficient to use a model for graceful degradation with simple thresholds (see Section 3.2). In the statistical setting, it follows from the bound for general adversaries that incomparable maximal adversaries exist: 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, and we need to consider multi-thresholds (see Section 3.3).

For (multi-)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 *robustness parameter* E (corresponding to $\mathcal{R}$). Since we consider multi-thresholds, the robustness parameter E is a list of pairs (e_a, e_p) where $e_a \leq e_p$. For secrecy, it follows from the bound in Theorem 3 that the actively corrupted parties $\mathcal{D}^*$ are not relevant. Hence, there cannot be two incomparable maximal adversaries for secrecy, and it is sufficient to use a single threshold.

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.

5.3.1 A Parametrized Protocol for Threshold Adversaries

In the following, we present the parametrized subprotocols and analyze them with respect to correctness, secrecy, and robustness. We do not consider fairness. The main result (including fairness) is discussed in Section 5.3.2. The protocol is based on IC signatures as introduced in Section 5.1.

5.3.1.1 The Underlying Verifiable Secret Sharing Scheme

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

Definition 9 (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 (correct) 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 (correct) 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 also with $[s]_i$. A sharing parameter d is *t_p-secret* if the shares held by any set of at most t_p parties are statistically independent from the secret, and *t_a-permissive* if the shares of all but t_a parties uniquely define the secret.

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 28. *A d-sharing is t_p-secret if $t_p \leq d$, and t_a-permissive if $t_a < n - d$.*

Proof. It follows directly from the properties of a polynomial of degree d and the secrecy of IC-signatures that any set of at most d parties has no information about the secret. Furthermore, $t_a < n - d$ implies that there remain at least $d + 1$ 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 Figure 5.10). Note that, in contrast to the corresponding share protocol in the perfect setting (Figure 4.5), we preserve the second dimension of $g(x, y)$.

Lemma 29. *Let d be the sharing parameter. On input s from the dealer, SHARE_S^{TH} 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 $g'(x, 0)$ and $g'(\alpha_i, y)$ for all α_i are 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[16] (Figure 5.11) proceeds share-wise: For each share s_i, first party p_i broadcasts the share s_i together with the sharing

[16] Private reconstruction can be reduced to public reconstruction using a blinding technique as discussed in Section 2.3.3.

Protocol $\text{SHARE}_{\text{S}}^{\text{TH}}$: 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 $\hat{s}(x) := g(x, 0)$.

Figure 5.10: The statistically secure share protocol for threshold adversaries.

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 $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq (e_a, e_p)$ for some $(e_a, e_p) \in E$, 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$_{\mathrm{S}}^{\mathrm{TH}}$ is the only subprotocol that might abort. All other protocols abort only if they use PUBLIC RECONSTRUCTION$_{\mathrm{S}}^{\mathrm{TH}}$ as a subprotocol and the invocation thereof aborts. Therefore, it is sufficient to discuss agreement on abort only for this protocol.

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 30. *Given the sharing parameter d, the robustness parameter E, and a d-sharing $[s]$ of some value s,* PUBLIC RECONSTRUCTION$_{\mathrm{S}}^{\mathrm{TH}}$ *reconstructs s to all parties. The protocol is 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 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:

Protocol PUBLIC RECONSTRUCTION$_{\mathbf{S}}^{\mathbf{TH}}$: 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 $s_{ij} = \hat{s}_i(\alpha_j)$[17] 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.

Figure 5.11: The statistically secure public reconstruction protocol for threshold adversaries.

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.

[17] That means, the value is consistent with the polynomial $\hat{s}_i(y)$ broadcasted by the dealer in

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".

Second, 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 these two observations above that shares from honest parties are always accepted.

Furthermore, 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 passively corrupted parties are accepted (in addition to the shares from honest parties). Since $|\mathcal{D}^*| < n - d$ there are always at least $d + 1$ correct (honest or passively corrupted) parties and the protocol does not abort.

Since the abort decision is based only on broadcasted values, we always have AGREEMENT ON ABORT (cf. Section 2.3.2). □

5.3.1.2 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

Step 1(a).

value can be achieved by letting each party p_i share a random value r_i, and computing $[r] = [r_1] + \ldots + [r_n]$ (see Section 2.3.3). For the multiplication of two shared values, we first describe 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 (Figure 5.12, [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 $v_i = a_i b_i$ by invoking ABC-PROOF$_{\mathrm{S}}^{\mathrm{TH}}$ on share polynomials of a_i, b_i, and v_i. If this proof is not accepted, the non-robust multiplication protocol is aborted.

Protocol NR-MULTIPLICATION$_{\mathbf{S}}^{\mathbf{TH}}$: 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$_{\mathrm{S}}^{\mathrm{TH}}$ on v_i, resulting in $[v_i]$.
   (b) Invoke ABC-PROOF$_{\mathrm{S}}^{\mathrm{TH}}$ 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.

Figure 5.12: The statistically secure, non-robust multiplication protocol for threshold adversaries.

Lemma 31. *Given are the sharing parameter d, and d-sharings of a and b. If $2d < n$ and the subprotocols are correct against $(|\mathcal{D}^*|, |\mathcal{E}^*|)$,* NR-MULTIPLICATION$_{\mathrm{S}}^{\mathrm{TH}}$ *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$_S^{TH}$ subprotocol (Figure 5.13) 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].

Protocol ABC-PROOF$_S^{TH}$: 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$_S^{TH}$ 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".

Figure 5.13: The statistically secure protocol for proving that $c = ab$ for threshold adversaries.

Lemma 32. *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$_S^{TH}$ *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 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 (Figure 5.14): First, using *dispute control* [BH06], we repeatedly invoke NR-MULTIPLICATION$_S^{TH}$ 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 $(b - y)$.

Lemma 33. *Given the sharing parameter d, the robustness parameter E, and d-sharings of a and b,* MULTIPLICATION$_S^{TH}$ *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$_S^{TH}$ *is robust against $(|\mathcal{D}^*|, |\mathcal{E}^*|)$.*

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

1. Create sharings $[x]$ and $[y]$ of random values x and y.
2. Invoke NR-MULTIPLICATION$_S^{TH}$ on $[x]$ and $[y]$ to compute $[z]$ for $z = xy$.
3. If NR-MULTIPLICATION$_S^{TH}$ succeeded, then invoke PUBLIC RECONSTRUCTION$_S^{TH}$ on $[a - x]$ and $[b - y]$, and compute and output $[c] = (a - x)(b - y) + (a - x)[y] + (b - y)[x] + [z]$.
4. If NR-MULTIPLICATION$_S^{TH}$ 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 NR-MULTIPLICATION$_S^{TH}$. Then, each party retraces the execution of these steps locally to detect disputing parties.[18]
   (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.

Figure 5.14: The statistically secure, robust multiplication protocol for threshold adversaries.

Proof. The protocol is repeated until NR-MULTIPLICATION$_S^{TH}$ succeeds. In the repetitions where NR-MULTIPLICATION$_S^{TH}$ does not succeed, correctness and secrecy of $[x]$, $[y]$, and $[z]$ do not need to be maintained. In the repetition where NR-MULTIPLICATION$_S^{TH}$ 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. □

[18] 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.

5.3.1.3 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 $\pi_{\mathrm{S}}^{d,E}$:

Lemma 34. *Let d be the sharing parameter, and E be the robustness parameter, the protocol $\pi_{\mathrm{S}}^{d,E}$ guarantees* 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 secrecy *if additionally $|\mathcal{E}^*| \leq d$, and/or* robustness *if additionally $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq E$. It always guarantees* agreement on abort.

Proof. $\pi_{\mathrm{S}}^{d,E}$ provides a certain security guarantee against $(|\mathcal{D}^*|, |\mathcal{E}^*|)$ if all subprotocols (cf. Lemmas 29 to 33) and the sharing (cf. Lemma 28) provide this guarantee against $(|\mathcal{D}^*|, |\mathcal{E}^*|)$. For each guarantee, it can easily be verified that the condition in the lemma implies the corresponding conditions in the mentioned lemmas. □

5.3.2 Main Result

The following theorem states the optimal bound for statistically secure reactive MPC for multi-threshold adversaries with both mixed adversaries and hybrid security. We show that the bound is sufficient for reactive MPC by providing parameters for the generalized protocols described above. Necessity follows directly from the corresponding proof for general adversaries.

Theorem 4. *In the secure channels model with broadcast and multi-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*

$$\begin{aligned}
&\forall (t_a^c, t_p^c) \in T^c, (t_a^r, t_p^r) \in T^r, (\cdot, t_p^s), (\cdot, t_p^{s\prime}) \in T^s: \\
&\Big(t_p^s + t_p^{s\prime} < n \quad \wedge \quad t_p^s + t_a^c < n \quad \wedge \\
&\quad (t_a^c + t_a^r + t_p^s < n \vee (t_p^s + t_p^r < n \wedge t_a^c + t_p^r < n) \vee (t_p^s + t_p^c < n \wedge t_a^r + t_p^c < n)) \Big) \\
&\vee \quad T^s = \{(0,0)\}.
\end{aligned}$$

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

Proof. The necessity of the bound in the theorem follows directly from the corresponding proof for general adversaries. To prove the sufficiency of the bound, we first note that if $T^s = \{(0,0)\}$, then there is no secrecy requirement, and we can directly use the trivial non-secret protocol described in Appendix 4.1.2. Otherwise, we employ the parametrized protocol $\pi_{\mathrm{S}}^{d,E}$ described in Section 5.3.1.3 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 34 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. □

5.4 Summary

We have provided statistically secure MPC protocols with graceful degradation of both security and corruptions. As in the perfect setting, the protocols are strict generalizations of existing protocols in the literature. Furthermore, we prove the tightness of the bounds achieved by our protocols.

These results provide insights into the relations between passive corruption and different security requirements. In particular, the bounds 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, which is in contrast to the perfect setting. 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 in the perfect setting, we obtain a

novel tradeoff between active and passive corruptions even when only considering a single security guarantee.

Finally, we introduced Group Commitments as a new primitive, which might be of independent interest. This primitive allows a group to first commit to a value on which they agree (while providing secrecy with respect to the remaining parties), and then to reveal this value at a later point in time to the remaining parties. In this work, we show how information-theoretic signatures in VSS schemes can be replaced with this primitive such that also passively corrupted parties can reliably verify the revealed secret.

Chapter 6

Protocols with Computational Security

For the construction of our protocols with computational security, we use again the same approach as in the information-theoretical settings: We first design parametrized protocols and analyze them with respect to correctness, secrecy, fairness, and robustness. Then we deduce bounds on the security guarantees, and prove that these bounds are tight.

In [IKLP06], the authors present a protocol for reactive MPC in the active world that provides full security up to a first threshold t, and correctness and secrecy up to a second threshold s, given that $t < n/2$ and $s + t < n$. We extend this protocol with fairness and security against mixed adversaries, while at the same time removing the restriction that robustness can only be guaranteed against a corrupted minority.

In contrast to the information-theoretical settings, here we additionally consider non-reactive MPC. In their seminal paper [GMW87], the authors provide two different protocols, one for passive security against up to $t < n$ corruptions, and one for active security against up to $t < \frac{n}{2}$ corruptions. In [IKLP06], these two protocols are combined into a single protocol, which is secure against an adversary that either passively corrupts any number of parties or actively corrupts a minority of the parties. This combined protocol is only applicable for non-reactive functions, and it is proven that this combination is impossible for reactive MPC.

Here, we present an MPC protocol (for non-reactive functions) with a dynamic tradeoff between active and passive corruptions. As [IKLP06], the protocol provides the best possible security level in presence of a purely passive adversary (namely $t < n$) as well as in presence of a purely active adversary (namely $t < n/2$). In addition, the protocol also tolerates mixed adversaries that corrupt some parties actively and some other parties passively, as long as at most k parties are corrupted actively and at most $n - k - 1$ parties are corrupted in total. Note that k need *not* be known, as it is not a parameter of the protocol.

In order to construct the protocols for both reactive and non-reactive MPC, we introduce the notion of *gradual* verifiable secret sharing (VSS). In contrast to traditional VSS, a gradual VSS reduces the number of adversaries against which secrecy is guaranteed during reconstruction in a step-wise fashion, and at the same time increases the number of adversaries against which robustness is guaranteed. By that, if the reconstruction of a secret aborts, secrecy against many adversaries is still guaranteed.

In Section 6.1 we introduce the notion of gradual VSS. Then we provide protocols for both non-reactive and reactive MPC (Sections 6.3 and 6.4, respectively), each for both general and threshold adversaries. Our protocols are based on the protocol from [GMW87], which we outline in Section 6.2.

6.1 Gradual Verifiable Secret Sharing

We first briefly review the standard definition of verifiable secret sharing (VSS) schemes. Then, we define a new property for VSS schemes introducing the notion of gradual reconstruction.[19] Finally, we present schemes for both threshold and general adversaries that achieve the new requirements.

6.1.1 Definitions

The following definition captures the standard, well-known properties of verifiable secret sharing.

Definition 10 (VSS)**.** A *Verifiable Secret Sharing* (VSS) scheme allows a designated party (the *dealer*) to share a value s among all parties, such that these

[19] This notion should not be confused with the notion of gradual release of secrecy as introduced by [Blu83].

parties can jointly reconstruct the value. More formally, we say a pair of protocols SHARE and REC is a $(\mathcal{Z}^s, \mathcal{Z}^r)$-secure VSS if the following conditions are fulfilled:

SECRECY: If $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^s$, then in SHARE the adversary obtains no information about s.

CORRECTNESS: After SHARE, the dealer is bound to a value s'. If the dealer is correct, then $s' = s$. Furthermore, in REC, either each (correct) party outputs s' or all (correct) parties abort.

ROBUSTNESS: The adversary cannot abort SHARE. If $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^r$, then the adversary cannot abort REC.

For $(\mathcal{D}^*, \mathcal{E}^*) \notin \mathcal{Z}^r$, this definition does not rule out that the reconstruction protocol aborts even in an unfair way, where the honest parties do not learn the secret but the corrupted parties do. In fact, most VSS schemes in the literature show this undesired behavior: When corrupted parties do not broadcast their shares, still they learn the shares from the honest parties and can compute the secret, but the honest parties do not obtain enough shares and abort.

Clearly, a certain level of unfairness cannot be avoided when secrecy and robustness are to be guaranteed with respect to many corruptions. In particular, whenever a sharing scheme is secret with respect to some subset $M \subseteq \mathcal{P}$ of the parties, then it cannot be robust with respect to the complement $\mathcal{P} \setminus M$ of this subset: When the parties in M have no information about the shared value after SHARE, and the parties in $\mathcal{P} \setminus M$ do not participate in REC, then the value cannot be reconstructed. Hence, the collection of subsets against which a sharing is secret implicitly defines the collection of subsets that can abort reconstruction (namely, the complements). In usual reconstruction protocols, all correct parties directly broadcast their entire shares, i.e., secrecy is given up against all subsets at once, before robustness against a single subset is achieved. This means that during reconstruction, any subset of parties that can abort, can also abort in an unfair way. Our new definition below requires that the transition from secrecy to robustness is gradual, one subset at a time, where the order is specified by a list $\mathcal{L}$ of subsets of $\mathcal{P}$ against which secrecy is guaranteed after SHARE. Then, whenever secrecy against a certain subset is given up, robustness against the complement of this subset is immediately obtained. So while processing the i-th element of $\mathcal{L}$, secrecy is still guaranteed against all later subsets, and robustness is already gained against the complements of all earlier subsets. If the protocol aborts, all parties output the same pair (B, i). The intuition behind this output is that the reconstruction proceeds along the list $\mathcal{L}$, and was interrupted when revealing the secret to the

parties in the i-th subset $L_i \in \mathcal{L}$ by the parties in the complement of L_i ($= B$) not revealing the necessary information.[20]

Definition 11 (Gradual VSS)**.** A $(\mathcal{Z}^s, \mathcal{Z}^r)$-secure VSS is *gradual* with respect to a list $\mathcal{L}$ consisting of sets $\mathcal{E}^s$ from $\mathcal{Z}^s$ (i.e., $\forall L_i \in \mathcal{L} : (\emptyset, L_i) \in \mathcal{Z}^s$), if the following conditions are fulfilled: If REC aborts, each (correct) party outputs the same (B, i) with $B \neq \emptyset$ such that $B \subseteq \mathcal{D}^*$, $B \cup L_i = \mathcal{P}$, and the adversary obtained no information about the secret if $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^s$ and $\mathcal{E}^* \subseteq L_j$ for some $j > i$ (i.e., $\exists L_j \in \mathcal{L}, j > i : \mathcal{E}^* \subseteq L_j$).[21]

Threshold Adversaries. These definitions can also be applied to (multi-) threshold adversaries by replacing the adversary structures $\mathcal{Z}^s$ and $\mathcal{Z}^r$ with (pairs of) thresholds (t_a^s, t_p^s) and (t_a^r, t_p^r), or more general, in a setting with incomparable maximal adversaries (see Section 3.3), with multi-thresholds T^s and T^r (i.e., secrecy of SHARE is required if $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq T^s$, and robustness of REC if $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq T^r$). However, it turns out that the definition of a gradual VSS can be simplified. In the threshold setting considered here, only the number of *passively* corrupted parties influences secrecy. Hence, there is a natural ordering of the adversaries, namely in decreasing size of t_p^s, and we do not need an extra parameter for the ordering (i.e., we can omit the list $\mathcal{L}$). Instead of an index i, the protocol outputs a bound t stating directly the maximal number of passively corrupted parties against which secrecy is still maintained.

Definition 12 (Gradual threshold VSS)**.** A (T^s, T^r)-secure VSS is *gradual* if the following conditions are fulfilled: If REC aborts, each party outputs (B, t) with $B \neq \emptyset$ such that $B \subseteq \mathcal{D}^*$, $|B| + t + 1 \geq n$, and the adversary obtained no information about the secret if $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq T^s$ and $|\mathcal{E}^*| \leq t$.

6.1.2 A Gradual VSS for General Adversaries

We describe a gradual VSS scheme for general adversaries. This scheme is based on [Mau02] extended with techniques from [HMZ08] and commitments (e.g. [Ped91]). Note that the VSS scheme in [HMZ08] requires an honest majority, which is not guaranteed in our setting. This construction results in the scheme $\text{VSS}_{\text{C}}^{\mathcal{S}} = (\text{SHARE}_{\text{C}}^{\mathcal{S}}, \text{REC}_{\text{C}}^{\mathcal{S}})$ which takes as input a list of subsets of

[20] In this step, the secret would have been revealed only to those parties in L_i that are not covered by a later subset $L_j, j > i$.

[21] That means, all corrupted parties are covered by some set against which secrecy was not yet given up.

$\mathcal{P}$, called sharing specification $\mathcal{S} = (S_1, \ldots, S_\ell)$ where $S_i \subseteq \mathcal{P}$. For a sharing of some value s, summands $s_1, \ldots, s_\ell$ with $s_1 + \ldots + s_\ell = s$ are chosen at random, and each party $p_j \in S_i$ obtains s_i and is committed to it.

Definition 13 ($\mathcal{S}$-sharing)**.** A value s is *$\mathcal{S}$-shared* for sharing specification $\mathcal{S} = (S_1, \ldots, S_\ell)$ where $S_i \subseteq \mathcal{P}$, denoted by $[s]$, if there are values $s_1, \ldots, s_\ell$, such that $s_1 + \ldots + s_\ell = s$ and, for all i, every (correct) party $p_j \in S_i$ holds the summand s_i, and is committed to it. A sharing specification $\mathcal{S}$ is *$\mathcal{E}$-secret* if the parties in $\mathcal{E}$ jointly cannot (efficiently) derive any information about the secret from their shares, and *$\mathcal{D}$-permissive* if the shares held by the parties in $\mathcal{P} \setminus \mathcal{D}$ uniquely define the secret.

If the sharing specification $\mathcal{S}$ is not clear from the context, we indicate it using the notation $[s]^{\mathcal{S}}$.

Lemma 35. *An $\mathcal{S}$-sharing is $\mathcal{E}$-secret if $\exists S_i \in \mathcal{S} : S_i \cap \mathcal{E} = \emptyset$, and $\mathcal{D}$-permissive if $\forall S_i \in \mathcal{S} : S_i \setminus \mathcal{D} \neq \emptyset$.*

Proof. Secrecy follows from the fact that $\mathcal{E}$ lacks at least one summand s_i, and from the hiding property of the commitment scheme. Furthermore, given that $\forall S_i \in \mathcal{S} : S_i \setminus \mathcal{D} \neq \emptyset$, each summand s_i is held by at least one party in $\mathcal{P} \setminus \mathcal{D}$. Hence, the secret s is uniquely defined by $s = s_1 + \ldots + s_\ell$. □

For the share protocol (Figure 6.1), we assume that commitments can be transferred by sending the opening information. This protocol is a straightforward adaption of the protocol in [Mau02, HMZ08].

Lemma 36. *Given a sharing specification $\mathcal{S}$ and input s, $\text{SHARE}_C^{\mathcal{S}}$ correctly, secretly and robustly computes an $\mathcal{S}$-sharing $[s]$. If $\mathcal{S}$ is $\mathcal{D}^*$-permissive, and if the dealer is correct, the sharing uniquely defines the secret s.*

Proof. CORRECTNESS: First, we have to show that the protocol outputs a valid $\mathcal{S}$-sharing. Since the commitments are binding, all correct parties $p_j \in S_i$ open c_i to the same value in Step 2. Hence, the view of all correct parties is consistent, which implies a valid 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. This follows immediately from the fact that a correct dealer always responds on reported inconsistencies with the original summands. Hence, the unique value defined by the sharing is the secret s.

SECRECY: Given a correct dealer, a valid $\mathcal{S}$-sharing is distributed in the first step. In the remaining protocol run, no additional information is revealed to

Protocol $\text{SHARE}_{\text{C}}^{\mathcal{S}}$: 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, he computes a commitment c_i together with opening information o_i, sends o_i to every party $p_j \in S_i$, and broadcasts c_i.
2. For each $S_i \in \mathcal{S}$: Every party $p_j \in S_i$ broadcasts a complaint bit, indicating whether o_i opens c_i to some value s_i'.
3. For each share s_i for which an inconsistency was reported, the dealer broadcasts the opening information o_i, and if o_i opens c_i, all parties in S_i accept o_i. Otherwise, the dealer is disqualified (and a default sharing of a default value is used).
4. Each party p_j outputs its share $\{(s_i, o_i) \mid p_j \in S_i\}$ and all commitments.

Figure 6.1: The computationally secure share protocol for general adversaries.

the adversary: A summand s_i is broadcasted in Step 3 only if a party p_j with $p_j \in S_i$ reported an inconsistency. Yet, such an inconsistency occurs only if one of the parties in S_i (or the dealer) is actively corrupted, i.e., when the adversary knew the value already beforehand.

ROBUSTNESS: It follows from inspection that the protocol cannot be aborted. □

This share protocol provides resilience even against a corrupted dealer. It turns out that in our protocols, essentially only ideal functionalities need to compute $\mathcal{S}$-sharings. Trivially, given a value s, such an honest dealer can directly sample and distribute a correct sharing $[s]$ without running $\text{SHARE}_{\text{C}}^{\mathcal{S}}$. The (probabilistic) function that samples shares of some given input s is denoted by $\text{STATE}_{\text{C}}^{\mathcal{S}}$.

In Figure 6.2, we describe the reconstruction protocol for a single sharing. Clearly, this protocol can be extended to reconstruct multiple sharings in parallel by executing the protocol on a vector of sharings, where an abort in one instance implies an immediate abort (in the same round) for all instances.

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

1. Reconstruct each summand s_i one after another according to the ordering in $\mathcal{S}$:
   (a) Each party $p_j \in S_i$ opens the commitment to s_i via broadcast.
   (b) If at least one party correctly opened the commitments to its respective share, the value of the party with the smallest index that opened correctly is used. If no party opened the commitment correctly, the protocol is aborted and each party outputs (B, i) with $B = S_i$.
2. Each party outputs the secret $s = s_1 + \ldots + s_\ell$.

Figure 6.2: The computationally secure protocol for gradual reconstruction for general adversaries.

Lemma 37. *Given is an $\mathcal{S}$-sharing $[s]$ with $\emptyset \notin \mathcal{S}$. If $\forall S_i \in \mathcal{S} : S_i \not\subseteq \mathcal{D}^*$, then $\text{REC}_{\text{C}}^{\mathcal{S}}$ (robustly) outputs s to all parties. Otherwise, either it outputs s to all parties, or it aborts and outputs (B, i) with $B \neq \emptyset$ such that $B \subseteq \mathcal{D}^*$, $S_i \subseteq B$, and the adversary obtained no information about the secret if $\exists S_j \in \mathcal{S}, j > i : \mathcal{E}^* \cap S_j = \emptyset$.*

Proof. CORRECTNESS: The only operation in the protocol is the opening of commitments. Hence, given a correct sharing and the binding property of the commitment scheme, incorrect parties cannot deviate without being detected.

ROBUSTNESS: To abort the reconstruction of some s_i, all parties $p_j \in S_i$ must refuse to correctly open their commitments. Since $\forall S_i \in \mathcal{S} : S_i \not\subseteq \mathcal{D}^*$, there is at least one correct party that correctly opens its commitment. Hence, the protocol does not abort.

GRADUAL: The reconstruction aborts with (B, i) only if the reconstruction of s_i failed. In that case, all parties in S_i refused to correctly open their commitments, and therefore $B = S_i \subseteq \mathcal{D}^*$. It follows from inspection that $S_i \subseteq B$, and since $\emptyset \notin \mathcal{S}$, we have $B \neq \emptyset$. Furthermore, if $\exists S_j \in \mathcal{S}, j > i : \mathcal{E}^* \cap S_j = \emptyset$, the adversary has no information about s_j (and hence about s) since the reconstruction of s_j did not yet start. □

The following corollary summarizes Lemma 35 ($\mathcal{S}$-sharing), Lemma 36 ($\text{SHARE}_{\text{C}}^{\mathcal{S}}$) and Lemma 37 ($\text{REC}_{\text{C}}^{\mathcal{S}}$):

Corollary 3. *Given a sharing specification $\mathcal{S}$ with $\emptyset \notin \mathcal{S}$, $\text{VSS}_{\text{C}}^{\mathcal{S}}$ is a $(\mathcal{Z}^s, \mathcal{Z}^r)$-secure, gradual (with respect to a list $\mathcal{L}$) VSS where*

- *$(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^s$ if $\exists S_i \in \mathcal{S} : S_i \cap \mathcal{E}^* = \emptyset$,*
- *$(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^r$ if $\forall S_i \in \mathcal{S} : S_i \not\subseteq \mathcal{D}^*$, and*
- *$\mathcal{L} = \overline{\mathcal{S}}$ (with $\overline{\mathcal{S}} = (\overline{S_1}, \ldots, \overline{S_l})$ for $S_i \in \mathcal{S}$).*

As a matter of course, given a list $\mathcal{L}$ and a secrecy structure $\mathcal{Z}^s$, it is straightforward to derive a sharing specification $\mathcal{S}$ such that the resulting scheme is gradual with respect to $\mathcal{L}$ and provides secrecy against adversaries from $\mathcal{Z}^s$: Select the (maximal[22]) sets $\mathcal{E}^s$ from $\mathcal{Z}^s$, and prepend these sets to $\mathcal{L}$. Then, choose $\mathcal{S}$ to be the component-wise complement of this extended list.

6.1.3 A Gradual VSS for Threshold Adversaries

We describe a gradual VSS scheme for threshold adversaries based on the standard Shamir sharing scheme [Sha79], and extended with (homomorphic) commitments to provide verifiability (e.g. [Ped91]). To obtain the gradual property, summands $s_1, \ldots, s_d$ with $s_1 + \ldots + s_d = s$ are chosen at random and, rather than the secret itself, these summands are shared, where summand s_i is shared with degree i. Then, during reconstruction, the summands are reconstructed one by one, in decreasing order of the sharing degree. We assume that each party p_i is assigned a unique and publicly known evaluation point $\alpha_i \in \mathbb{F} \setminus \{0\}$,[23] and that the commitments are homomorphic and transferable by sending the opening information. This construction results in the scheme $\text{VSS}_{\text{C}}^d = (\text{SHARE}_{\text{C}}^d, \text{REC}_{\text{C}}^d)$ for parameter d.

Definition 14 (d-sharing)**.** A value s is *d-shared*, denoted by $[s]$, if there are values $s_1, \ldots, s_d$, such that $s_1 + \ldots + s_d = s$ and, for all $i \in \{1, \ldots, d\}$, there is a polynomial $\hat{g}_i(x)$ of degree i with $\hat{g}_i(0) = s_i$, and every (correct) party p_j holds a share $s_{ij} = \hat{g}_i(\alpha_j)$ and is committed to it. A sharing degree d is *t_p-secret* if up to t_p parties jointly cannot (efficiently) derive any information about the secret from their shares, and *t_a-permissive* if the shares of all but t_a parties uniquely define the secret.

Lemma 38. *A d-sharing is t_p-secret if $t_p \leq d$, and t_a-permissive if $t_a < n - d$.*

[22] A set $\mathcal{E}^s$ is maximal in $\mathcal{Z}^s$ if there is no other set $\mathcal{E}^{s\prime}$ in $\mathcal{Z}^s$ with $\mathcal{E}^s \subset \mathcal{E}^{s\prime}$ (cf. Section 2.2.2). Note that it is not mandatory to choose only maximal sets here.

[23] This implies that the field $\mathbb{F}$ must have more than n elements.

Proof. The summand s_d is shared with degree d. Therefore, it follows directly from the properties of a polynomial of degree d and the hiding property of the commitments that any set of at most d parties has no information about the secret. Furthermore, $t_a < n - d$ implies that there remain at least $d + 1$ parties whose shares uniquely define a share polynomial. □

The sharing protocol from [Ped91] can be extended in a straightforward way to compute such a d-sharing. For this protocol (Figure 6.3), we assume that commitments can be transferred by sending the opening information.

Protocol $\text{SHARE}_{\text{C}}^{d}$: Given input s from the dealer, compute a d-sharing of this value.

1. The dealer chooses uniformly random summands $s_1, \ldots, s_d$ with $\sum_{i=1}^{d} s_i = s$.
2. For $i \in \{1, \ldots, d\}$:
   (a) The dealer chooses a random polynomial $g_i(x)$ of degree i with $g_i(0) = s_i$, and computes and broadcasts (homomorphic) commitments of the coefficients of $g_i(x)$.
   (b) For each share $s_{ij} = g_i(\alpha_j)$, each party locally computes a commitment c_{ij} (using the homomorphic property), and the dealer sends the corresponding opening information o_{ij} to party p_j. Then, p_j broadcasts a complaint bit, indicating whether o_{ij} opens c_{ij} to some value s'_{ij}.
   (c) For each share s_{ij} for which an inconsistency was reported, the dealer broadcasts the opening information o_{ij}, and if o_{ij} opens c_{ij}, p_j accepts o_{ij}. Otherwise, the dealer is disqualified (and a default sharing of a default value is used).
3. Each party p_j outputs its share $\big((s_{1j}, o_{1j}), \ldots, (s_{dj}, o_{dj})\big)$ and all commitments.

Figure 6.3: The computationally secure share protocol for threshold adversaries.

Lemma 39. *Given a parameter d and input s, $\text{SHARE}_{\text{C}}^{d}$ correctly, secretly, and robustly computes a d-sharing $[s]$. If d is $|\mathcal{D}^*|$-permissive, and if the dealer is correct, the sharing uniquely defines the secret s.*

Proof. CORRECTNESS: First, we have to show that the protocol outputs a valid d-sharing. Trivially, in Step 2.a, any (well-formed) commitments broadcasted by the dealer are correct. In Step 2.b, commitments to all shares are computed locally by each party directly from the commitments to the coefficients broadcasted in Step 2.a. Hence, all (correct) parties have a consistent view with correct commitments. In Steps 2.b and 2.c, due to the binding property of the commitments, the adversary cannot distribute inconsistent opening information without being detected. Hence, the sharing is correct (or the dealer is disqualified and a default sharing is used). 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 with the correct values. Hence, if d is $|\mathcal{D}^*|$-permissive, the unique value defined by the sharing is the secret s.

SECRECY: The commitments are computationally hiding. Therefore, the adversary obtains no information in Step 2.a of $\text{SHARE}_{\text{C}}^{d}$. Furthermore, the opening information distributed in Step 2.b corresponds to a valid d-sharing. Finally, in Step 2.c, the adversary obtains no additional information: Whenever a value is broadcasted, the adversary knew this value already beforehand.

ROBUSTNESS: By inspection, the share protocol does not abort. □

The share protocol provides resilience even against a corrupted dealer. It turns out that in our protocols, essentially only ideal functionalities need to compute d-sharings. Trivially, given a value s, such an honest dealer can directly sample and distribute a correct sharing $[s]$ without running $\text{SHARE}_{\text{C}}^{d}$. The (probabilistic) function that samples shares of some given input s is denoted by $\text{STATE}_{\text{C}}^{d}$.

The reconstruction protocol for a single sharing is described in Figure 6.4. As in the setting with general adversaries, the extension to multiple sharings is straightforward.

Lemma 40. *Given is a d-sharing $[s]$ for $d < n$. If $|\mathcal{D}^*| < n - d$, then $\text{REC}_{\text{C}}^{d}$ (robustly) outputs s to all parties. Otherwise, either it outputs s to all parties, or it aborts and outputs (B, t) with $B \neq \emptyset$ such that $B \subseteq \mathcal{D}^*$, $|B| + t + 1 \geq n$, and the adversary obtained no information about the secret if $|\mathcal{E}^*| \leq t$.*

Proof. CORRECTNESS: The only operation in the protocol is the opening of commitments. Hence, given a correct sharing and the binding property of the commitment scheme, incorrect parties cannot deviate without being detected.

Protocol $\text{REC}_{\text{C}}^{d}$: Given a d-sharing of some value s, reconstruct s to all parties.

1. For $i = d$ down to 1:
   (a) Each party p_j opens the commitment to its share s_{ij} via broadcast.
   (b) If at least $i+1$ parties correctly opened the commitments to their respective shares, each party locally interpolates $g_i(x)$ and computes $s_i = g_i(0)$. Otherwise, the protocol is aborted and each party outputs $t = i-1$ and the set B of parties that did not broadcast correct opening information.
2. Each party outputs $s = s_1 + \ldots + s_d$.

Figure 6.4: The computationally secure protocol for gradual reconstruction for threshold adversaries.

ROBUSTNESS: To abort the reconstruction of some s_i, at least $n - i \geq n - d$ parties must refuse to correctly open their respective commitments. Hence, for $|\mathcal{D}^*| < n - d$, the protocol is robust.

GRADUAL: The reconstruction aborts with (B, t) only if the reconstruction of s_{t+1} failed. In that case, strictly less than $t + 2$ parties opened their commitments correctly. Therefore, $|B| > n-(t+2)$, i.e. $|B| \geq n-(t+1)$. Furthermore, since $d < n$, we have that $t < n - 1$, and hence $|B| > 0$, i.e., $B \neq \emptyset$. Clearly, $B \subseteq \mathcal{D}^*$, since only active parties do not open their commitments correctly. Furthermore, if $|\mathcal{E}^*| \leq t$, the adversary has no information about s_t since the reconstruction of s_t did not yet start. □

The following corollary summarizes Lemma 38 (d-sharing), Lemma 39 ($\text{SHARE}_{\text{C}}^{d}$) and Lemma 40 ($\text{REC}_{\text{C}}^{d}$):

Corollary 4. *Given a parameter $d < n$, $\text{VSS}_{\text{C}}^{d} = (\text{SHARE}_{\text{C}}^{d}, \text{REC}_{\text{C}}^{d})$ is a (T^s, T^r)-secure, gradual VSS where $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq T^s$ if $|\mathcal{E}^*| \leq d$, and $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq T^r$ if $|\mathcal{D}^*| < n - d$.*

6.2 The Protocol of [GMW87]

Our results are based on the protocol from [GMW87].[24] We briefly review the protocol description. For details, we refer the interested reader to the original work. The protocol is constructed in two steps: A first protocol provides security only against passive adversaries. Then, in a second step, a compiler is used to make the passive protocol secure also against active adversaries.

6.2.1 The Passive Protocol

The protocol proceeds in three phases: input, computation (addition and multiplication), and output. During the input phase, each party shares its input with an n-out-of-n XOR-sharing. This sharing is linear, hence additions and linear functions can be computed locally. For multiplication, a protocol for oblivious transfer is used. Finally, to reconstruct the output, each party broadcasts its corresponding share.

This protocol provides secrecy against $t < n$ passively corrupted parties. It is crucial for the secrecy of the protocol that all parties choose their randomness according to the protocol description. Otherwise, a party can break secrecy during the protocol for oblivious transfers. In contrast, correctness and robustness are independent from the choice of the randomness.

The original protocol considers only Boolean circuits, i.e., each party provides its input as single bits. However, any ideal functionality can be converted into a Boolean circuit in a straightforward way.

6.2.2 The Active Protocol

To make the above passive protocol secure also against active corruptions, it is modified using a compiler. A first, simpler version of the compiler provides only security with abort for $t < n$ actively corrupted parties (i.e., correctness and secrecy against $t < n$ corrupted parties, and, in case of an abort, each correct party outputs the same non-empty set $B \subseteq \mathcal{D}^*$). A second, extended version of the compiler provides a trade-off between secrecy and robustness.

[24] [GMW87] provides only stand-alone security. If UC security is required, the protocol of [CLOS02] can be used analogously.

6.2.2.1 The Compiler for Security with Abort.

The idea behind the first compiler is to force all parties to follow the protocol description using commitments and zero-knowledge proofs: Before the protocol starts, each party commits to its input. Furthermore, a coin-toss protocol is carried out among all parties to generate a random tape (to be used in the passive protocol) for each party in such a way, that the party is also committed to it.

Then, during the protocol execution, for each message, the sender has to attach a zero-knowledge proof that the messages is correct with respect to its inputs, randomness, and the messages received so far. For this purpose, all messages have to be broadcasted such that they are known to and can be checked by all parties. Secrecy of messages is guaranteed using public-key encryption.

We denote this protocol by `GMW`. As stated in [GMW87], it provides security with abort for $t < n$ corrupted parties. However, it can easily be seen that correctness (but not secrecy) can also be achieved for $t = n$ corrupted parties. In particular this holds also in the setting with mixed adversaries where some parties are actively and all remaining parties are passively corrupted. This follows from the fact that each party has to prove the correctness of the messages it sends using a zero-knowledge protocol. Given instant randomness (i.e., randomness generated only when needed, cf. Section 2.2.1), even the challenges of passively corrupted parties are unpredictable to the adversary (note that these challenges are used in the compiler, not in the passive protocol, and are hence not part of the predetermined random tape). If all parties are passively corrupted (and at least one party even actively), the adversary can choose all random tapes (for the passive protocol) during their generation before the protocol execution. As mentioned above, this does not affect correctness (or robustness). Concerning agreement on abort, note that the abort decision is based only on broadcasted values. Hence, either all correct parties abort, or all correct parties continue (cf. Section 2.3.2).

The fact that secrecy cannot be achieved if all parties are passively corrupted can be shown e.g. by means of plaintext-aware encryption [BR94, BP04]: Plaintext-awareness guarantees that no efficient algorithm can compute a valid ciphertext without being aware of the corresponding plaintext. To prove impossibility of MPC in a setting where all parties are passively corrupted, consider the functionality that samples a public-secret-key pair (pk, sk) (for a plaintext-aware encryption scheme), encrypts a random message $c = \mathrm{Enc}(pk, m)$, and outputs (pk, c). Assume that there is a protocol

that implements this functionality and provides secrecy against any number of corrupted parties. That means, in particular, the adversary obtains no information about the message m. Now, an algorithm could (locally) simulate such a protocol execution and output (pk, c) without being aware of m. This is a contradiction. Hence, under any assumption that implies the existence of plaintext aware encryption (e.g., the knowledge of exponent assumption [Den06]), secret MPC cannot be achieved when all parties are passively corrupted.

6.2.2.2 The Compiler for Full Security.

The above compiler does not provide robustness, not even against a single corrupted party. In the second compiler, to achieve a certain level of robustness, each party additionally shares its input using a $(\mathcal{Z}^s, \mathcal{Z}^r)$-secure VSS (thereby sacrificing some secrecy). Furthermore, the random tape (for the passive protocol) for each party is jointly generated such that it is also shared according to the VSS. As soon as a party deviates during the protocol execution, the inputs and the random tape of this party are reconstructed, and each party locally simulates the aborted party.

The original description in [GMW87] considers only VSS with a threshold of $n/2$. However, it is easy to see that any VSS can be used. The resulting protocol inherits the robustness and secrecy properties of the corresponding VSS, while leaving the correctness properties unchanged (the same holds for the simplified protocol in [Gol04, p. 735]).

This construction provides correctness for any number of corrupted parties, secrecy if $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^s$, and robustness if $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^r$. Furthermore, if the protocol is aborted, then each party outputs the same non-empty set $B \subseteq \mathcal{D}^*$. This implies in particular that we always have agreement on abort.

This protocol can be used for both threshold and general adversaries, depending on the VSS it is instantiated with. When we instantiate this protocol using $\text{VSS}_{\text{C}}^{\mathcal{S}}$ (Section 6.1.2), we denote the resulting protocol by $\texttt{GMW}^{\mathcal{S}}$, and when using $\text{VSS}_{\text{C}}^{d}$ (Section 6.1.3) with $\texttt{GMW}^{d}$. Note that for this extension of `GMW`, a standard, non-gradual VSS would be sufficient.

6.3 Non-Reactive Multi-Party Computation

Our protocol for non-reactive MPC for both threshold and general adversaries is based on an idea from [IKLP06]: Given the function f and the inputs

$x_1, \ldots, x_n$, the protocol first distributedly computes $y = f(x_1, \ldots, x_n)$ using a correct and secret, but non-robust MPC protocol. Yet, instead of y itself, this MPC protocol outputs a sharing of y that was computed according to some VSS scheme. Then, the parties reconstruct this sharing.

In [IKLP06], two (maximal) adversaries are considered: One, that actively corrupts a minority of the parties, and one that passively corrupts any number of parties. The output y is shared with a VSS for a threshold $t < n/2$. Then, whenever the non-robust MPC protocol aborts (i.e., the active adversary is present), the computation of $y = f(x_1, \ldots, x_n)$ is repeated with a robust MPC protocol, which provides security against an actively corrupted minority. Yet, in a more general setting with mixed adversaries, where a majority of the parties might be passively, and in addition some parties even actively corrupted, this solution becomes insecure: The adversary can abort the non-robust protocol during the output phase. Then, he might already have learned y, and repeating the computation would violate security.

In contrast, by using a gradual VSS scheme to share y (cf. Section 6.1), our protocols achieve stronger security guarantees. Given a set of actually (actively) cheating parties, a gradual VSS allows to maintain as much secrecy as possible. Then, in case of an abort during the reconstruction, the cheaters are identified and eliminated, and if the gradual VSS still guarantees enough secrecy,[25] the computation of $y = f(x_1, \ldots, x_n)$ is repeated using again the same MPC protocol among the remaining parties. Otherwise, the execution halts.

We use `GMW` (see Section 6.2.2.1) to implement the ideal functionality computing f and then a sharing of the result y. This protocol provides correctness for up to $t = n$, and secrecy for $t < n$ corrupted parties. Due to these specific security guarantees, this protocol can be used for both threshold and general adversaries.

6.3.1 Non-Reactive MPC for General Adversaries

In the setting with general adversaries, we use the gradual VSS scheme introduced in Section 6.1.2 for a given sharing specification $\mathcal{S}$. In fact, we only require the reconstruction protocol $\text{REC}_{\text{C}}^{\mathcal{S}}$ and the (probabilistic) function $\text{STATE}_{\text{C}}^{\mathcal{S}}$ that, given a value y, samples shares of y according to $\text{VSS}_{\text{C}}^{\mathcal{S}}$.

[25] The protocols are described with respect to a robustness parameter rather than a secrecy parameter as suggested here. It turns out that this simplifies the description and the proof.

Furthermore, the protocol receives as parameter a (monotone) robustness parameter $\mathcal{R} \subseteq 2^{\mathcal{P}}$ indicating which subsets of actively corrupted parties the protocol can eliminate (and then repeat the run) without violating security. A party is eliminated by removing the party from $\mathcal{P}$, and from all entries in $\mathcal{S}$ and $\mathcal{R}$.

Protocol for non-reactive MPC: Given is a function f.

1. Employ GMW to first compute $y = f(x_1, \ldots, x_n)$, where x_i is the input from party p_i, then evaluate $\text{STATE}_{\text{C}}^{\mathcal{S}}$ on y, and finally output to each party its corresponding share, resulting in $[y]$. If GMW aborts with a set B of active cheaters, repeat with $\mathcal{P} = \mathcal{P} \setminus B$ and $\mathcal{S}$ and $\mathcal{R}$ reduced accordingly.
2. Invoke $\text{REC}_{\text{C}}^{\mathcal{S}}$ on $[y]$. On abort with a set B of active cheaters: If $B \in \mathcal{R}$, then repeat the whole protocol with $\mathcal{P} = \mathcal{P} \setminus B$ and $\mathcal{S}$ and $\mathcal{R}$ reduced accordingly. Otherwise, halt the execution.
3. Output y.

Figure 6.5: The computationally secure protocol for non-reactive MPC for general adversaries.

Lemma 41. *Given a function f, a sharing specification $\mathcal{S}$ with $\emptyset \notin \mathcal{S}$, and a robustness parameter $\mathcal{R}$, the protocol for non-reactive MPC computes f in presence of an adversary corrupting $(\mathcal{D}^*, \mathcal{E}^*)$. The protocol always guarantees correctness and agreement on abort. Furthermore, it is robust if $\mathcal{D}^* \in \mathcal{R}$, secret if $\exists S_i \in \mathcal{S} : S_i \cap \mathcal{E}^* = \emptyset$ and*

$$\forall S_i \in \mathcal{S} : S_i \not\subseteq \mathcal{D}^* \ \vee \ \exists S_j \in \mathcal{S}, j > i : S_j \cap \mathcal{E}^* = \emptyset \ \vee \ S_i \notin \mathcal{R},$$

and fair if $\exists S_i \in \mathcal{S} : S_i \cap \mathcal{E}^* = \emptyset$ and

$$\forall S_i \in \mathcal{S} : S_i \not\subseteq \mathcal{D}^* \ \vee \ \exists S_j \in \mathcal{S}, j > i : S_j \cap \mathcal{E}^* = \emptyset.$$

Proof. CORRECTNESS follows trivially by inspection, and since the abort decision is based only on broadcasted values, we always have AGREEMENT ON ABORT (cf. Section 2.3.2).

ROBUSTNESS: Since $\mathcal{D}^* \in \mathcal{R}$ and $B \subseteq \mathcal{D}^*$, the protocol is repeated (with $\mathcal{P} = \mathcal{P} \setminus B$) until it succeeds. Since B is non-empty, the protocol is repeated at most n times.

SECRECY: Since $\exists S_i \in \mathcal{S} : S_i \cap \mathcal{E}^* = \emptyset$ (which in particular implies that $\mathcal{E}^* \neq \mathcal{P}$), both GMW and the output $[y]$ (Corollary 3) are secret. Hence, the

adversary obtains no information about the inputs, outputs, or intermediate results in Step 1. Given the output, Steps 2 and 3 are independent from the inputs and intermediate results. Therefore, if the protocol does not abort, secrecy is maintained. Yet, secrecy may be violated if the adversary can force a repetition of the protocol after learning the output.[26] If the protocol aborts in Step 2 with (B, i), we have that $B \subseteq \mathcal{D}^*$ and $B \cup L_i = \mathcal{P}$. Since $\mathcal{L} = \overline{\mathcal{S}}$ (Corollary 3), this implies $S_i \subseteq B \subseteq \mathcal{D}^*$. Then, it follows from the second part of the condition in the lemma that

$$\exists S_j \in \mathcal{S}, j > i : \; S_j \cap \mathcal{E}^* = \emptyset \vee S_i \notin \mathcal{R}.$$

If $\exists S_j \in \mathcal{S}, j > i : S_j \cap \mathcal{E}^* = \emptyset$, the adversary did not obtain any information about y (Corollary 3), and the protocol can be repeated without violating secrecy. Otherwise, if $S_i \notin \mathcal{R}$, then $B \notin \mathcal{R}$ and the protocol is aborted, i.e. the adversary learned at most one output value.

The proof of FAIRNESS is along the lines of the proof of secrecy, and is therefore omitted. □

Given Lemma 41, we can derive a tight bound for non-reactive MPC with general adversaries:

Theorem 5. *In the secure channels model with broadcast and general adversaries, computationally secure non-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*

$$\begin{aligned}
&\exists \text{ordering of } \mathcal{Z}^s : \\
&\Big(\quad \forall (\cdot, \mathcal{E}_i^s) \in \mathcal{Z}^s : \big(\forall (\mathcal{D}^r, \cdot) \in \mathcal{Z}^r : \; \mathcal{D}^r \cup \mathcal{E}_i^s \neq \mathcal{P} \big) \quad \vee \\
&\qquad\qquad\qquad\qquad \big(\forall (\mathcal{D}_k^s, \cdot) \in \mathcal{Z}^s, k \leq i : \mathcal{D}_k^s \cup \mathcal{E}_i^s \neq \mathcal{P} \big) \\
&\quad \wedge \; \forall (\mathcal{D}_k^f, \cdot) \in \mathcal{Z}^f, (\cdot, \mathcal{E}_i^s) \in \mathcal{Z}^s, k \leq i : \; \mathcal{D}_k^f \cup \mathcal{E}_i^s \neq \mathcal{P} \Big) \\
&\vee \quad \mathcal{Z}^s = \{(\emptyset, \emptyset)\}
\end{aligned}$$

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

Proof. To prove *sufficiency*, we first note that if $\mathcal{Z}^s = \{(\emptyset, \emptyset)\}$, there is no secrecy requirement, and we can directly use the trivial non-secret protocol described in Section 4.1.2. Otherwise, we use the protocol described in Figure 6.5 with $\mathcal{R} = \{R \mid (R, R) \in \mathcal{Z}^r\}$, and $\mathcal{S} = (\overline{\mathcal{E}_i^s} \mid (\emptyset, \mathcal{E}_i^s) \in \mathcal{Z}^s)$ (maintaining the order of $\mathcal{Z}^s$). It follows from the condition in the theorem that $\forall (\emptyset, \mathcal{E}_i^s) \in \mathcal{Z}^s : \mathcal{E}_i^s \neq \mathcal{P}$, and hence $\emptyset \notin \mathcal{S}$.

[26] In that case, the adversary may learn two evaluations of f for different inputs.

CORRECTNESS and AGREEMENT ON ABORT are always guaranteed, and ROBUSTNESS follows directly from the choice of $\mathcal{R}$.

SECRECY: Given is that $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^s$. It follows immediately from the choice of $\mathcal{S}$ that $\exists S_i \in \mathcal{S} : S_i \cap \mathcal{E}^* = \emptyset$. Furthermore, let i be an arbitrary index in $\mathcal{Z}^s$.

Case (1): $\forall (\mathcal{D}^r, \cdot) \in \mathcal{Z}^r : \mathcal{D}^r \cup \mathcal{E}_i^s \neq \mathcal{P}$. Given the choice of $\mathcal{S}$, we then have $\forall (\mathcal{D}^r, \cdot) \in \mathcal{Z}^r : S_i \not\subseteq \mathcal{D}^r$, and given the choice of $\mathcal{R}$, we have $S_i \notin \mathcal{R}$.

Case (2): $\forall (\mathcal{D}_k^s, \cdot) \in \mathcal{Z}^s, k \leq i : \mathcal{D}_k^s \cup \mathcal{E}_i^s \neq \mathcal{P}$. Since $\mathcal{Z}^s$ is monotone, there is an index j such that $(\mathcal{D}^*, \mathcal{E}^*) = (\mathcal{D}_j^f, \mathcal{E}_j^f)$. We distinguish whether $j \leq i$ or $j > i$: If $j \leq i$, then it follows from the assumption that $\mathcal{P} \setminus \mathcal{E}_i^s \not\subseteq \mathcal{D}^*$, i.e., given the choice of $\mathcal{S}$, that $S_i \not\subseteq \mathcal{D}^*$. If $j > i$, then given the choice of $\mathcal{S}$ we have that $\exists S_j \in \mathcal{S}, j > i : S_j \cap \mathcal{E}^* = \emptyset$.

FAIRNESS: The ordering of $\mathcal{Z}^s$ implies an ordering of $\mathcal{Z}^f$. Hence, this proof is along the lines of the proof of secrecy, and is therefore omitted.

Now, we prove that the bound in the theorem is also *necessary*, i.e. if violated, (non-reactive) MPC is impossible. The proof is inspired by [HMZ08]. The bound in the theorem is violated if

$$
\begin{aligned}
&\forall \text{orderings of } \mathcal{Z}^s : \\
&\Big(\quad \exists (\cdot, \mathcal{E}_i^s) \in \mathcal{Z}^s : \big(\exists (\mathcal{D}^r, \cdot) \in \mathcal{Z}^r : \mathcal{D}^r \cup \mathcal{E}_i^s = \mathcal{P}\big) \quad \wedge \\
&\qquad\qquad \big(\exists (\mathcal{D}_k^s, \cdot) \in \mathcal{Z}^s : k \leq i \wedge \mathcal{D}_k^s \cup \mathcal{E}_i^s = \mathcal{P}\big) \qquad (1) \\
&\quad \vee \exists (\mathcal{D}_k^f, \cdot) \in \mathcal{Z}^f, (\cdot, \mathcal{E}_i^s) \in \mathcal{Z}^s : k \leq i \wedge \mathcal{D}_k^f \cup \mathcal{E}_i^s = \mathcal{P}\Big) \qquad (2) \\
&\wedge \quad \mathcal{Z}^s \neq \{(\emptyset, \emptyset)\}
\end{aligned}
$$

Case (1). We first consider the case where $\mathcal{Z}^s \neq \{(\emptyset, \emptyset)\}$ and $\forall \text{orderings of } \mathcal{Z}^s, \exists (\cdot, \mathcal{E}_i^s) \in \mathcal{Z}^s :$

$$\big(\exists (\mathcal{D}^r, \cdot) \in \mathcal{Z}^r : \mathcal{D}^r \cup \mathcal{E}_i^s = \mathcal{P}\big) \quad \wedge \quad \big(\exists (\mathcal{D}_k^s, \cdot) \in \mathcal{Z}^s : k \leq i \wedge \mathcal{D}_k^s \cup \mathcal{E}_i^s = \mathcal{P}\big).$$

We can assume that $\mathcal{D}^r \subseteq \mathcal{D}_k^s$ (due to monotonicity), and that all three sets are non-empty: If $\mathcal{D}_k^s = \emptyset$ or $\mathcal{D}^r = \emptyset$, we have $\mathcal{E}_i^s = \mathcal{P}$, and if $\mathcal{E}_i^s = \emptyset$, we have $\mathcal{D}_k^s = \mathcal{E}_k^s = \mathcal{P}$. In both cases, MPC is impossible as stated in Section 6.2.2.1. Assume there is a protocol for such non-empty $\mathcal{D}^r$, $\mathcal{D}_k^s$ and $\mathcal{E}_i^s$. Every protocol defines an ordering in which the sets $\mathcal{E}^s$ (for $(\cdot, \mathcal{E}^s) \in \mathcal{Z}^s$) learn the result.[27] The assumption states that for every ordering of $\mathcal{Z}^s$, there are indices $k \leq i$ such that $\mathcal{D}_k^s \cup \mathcal{E}_i^s = \mathcal{P}$ and $\mathcal{D}^r \cup \mathcal{E}_i^s = \mathcal{P}$. Assume that the parties in $\mathcal{E}_k^s$ learn

[27] I.e., when the parties in $\mathcal{E}^s$ jointly hold enough information to be able to efficiently reconstruct the result.

the result in round ℓ. Now, the adversary corrupts $(\mathcal{D}^*, \mathcal{E}^*) = (\mathcal{D}_k^s, \mathcal{E}_k^s)$, and has the parties in $\mathcal{D}^r \subseteq \mathcal{D}_k^s$ stop sending any messages in and after round ℓ. The remaining parties are covered by $\mathcal{E}_i^s$. Hence, after round $\ell - 1$ they do not yet hold enough information to efficiently reconstruct the result (i.e., the state of the computation is lost). Yet, the adversary does learn the result. Since robustness (and correctness) must be guaranteed against $\mathcal{D}^r$, the protocol execution has to be repeated. However, it cannot be guaranteed that the parties in $\mathcal{D}^r$ provide the same input (e.g., their inputs might be replaced with default values). In that case, the adversary learns the results for two evaluations of the function, which constitutes a violation of secrecy.

As an example, consider the following (generalized OT-) functionality: Each party p_i inputs three bits: $a_0^{(i)}$, $a_1^{(i)}$, and $b^{(i)}$ (with default input $a_0^{(i)} = a_1^{(i)} = b^{(i)} = 0$). Let $d = b^{(1)} \oplus \ldots \oplus b^{(n)}$. The output is $y = (a_d^{(1)}, \ldots, a_d^{(n)})$. The adversary lets one actively corrupted party input $b = 1$, and all others $b = 0$. Then, with the attack described above, the adversary learns both $y_0 = (a_0^{(1)}, \ldots, a_0^{(n)})$ and $y_1 = (a_1^{(1)}, \ldots, a_1^{(n)})$, which clearly is a violation of secrecy.

Case (2). Now we consider the case that $\mathcal{Z}^s \neq \{(\emptyset, \emptyset)\}$ and

$$\forall \text{orderings of } \mathcal{Z}^s, \exists (\mathcal{D}_k^f, \cdot) \in \mathcal{Z}^f, (\cdot, \mathcal{E}_i^s) \in \mathcal{Z}^s : \; k \leq i \; \wedge \; \mathcal{D}_k^f \cup \mathcal{E}_i^s = \mathcal{P}.$$

We can assume that the sets $\mathcal{D}_k^f$ and $\mathcal{E}_i^s$ are non-empty: If $\mathcal{D}_k^f = \emptyset$, we have $\mathcal{E}_i^s = \mathcal{P}$, and if $\mathcal{E}_i^s = \mathcal{P}$, we have $\mathcal{D}_k^f = \mathcal{P}$, i.e., $\mathcal{E}^s = \mathcal{P}$ for some $\mathcal{E}^s$. In both cases, MPC is impossible as stated in Section 6.2.2.1. Assume there is a protocol for such non-empty $\mathcal{D}_k^f$ and $\mathcal{E}_i^s$. Every protocol defines an ordering in which the sets $\mathcal{E}^s$ (for $(\cdot, \mathcal{E}^s) \in \mathcal{Z}^s$) learn the result.[28] The assumption states that for every ordering of $\mathcal{Z}^s$ (which is also an ordering for $\mathcal{Z}^f \subseteq \mathcal{Z}^s$), there are indices $k \leq i$ such that $\mathcal{D}_k^f \cup \mathcal{E}_i^s = \mathcal{P}$. Assume that the parties in $\mathcal{E}_k^f$ learn the result in round ℓ. Now, the adversary corrupts $(\mathcal{D}^*, \mathcal{E}^*) = (\mathcal{D}_k^f, \mathcal{E}_k^f)$, and has the parties in $\mathcal{D}_k^f$ stop sending any messages in and after round ℓ. The remaining parties are covered by $\mathcal{E}_i^s$. Hence, after round $\ell - 1$ they do not yet hold enough information to efficiently reconstruct the result, while the adversary does learn the result. This constitutes a violation of fairness. □

6.3.2 Non-Reactive MPC for Threshold Adversaries

In the setting with threshold adversaries, we use the gradual VSS scheme described in Section 6.1.3 with degree $d = n - 1$. In fact, we only require

[28] I.e., when the parties in $\mathcal{E}^s$ jointly hold enough information to be able to efficiently reconstruct the result.

the reconstruction protocol $\text{REC}_{\text{C}}^{n-1}$ and the (probabilistic) function $\text{STATE}_{\text{C}}^{n-1}$ that, given a value y, samples shares of y according to $\text{VSS}_{\text{C}}^{n-1}$. Furthermore, the protocol receives a robustness parameter e stating the number of actively corrupted parties that the protocol can eliminate (and then repeat the run) without violating security. A set of parties is eliminated by removing the parties from $\mathcal{P}$ and reducing n and e accordingly.

Protocol for non-reactive MPC: Given is a function f.

1. Employ GMW to first compute $y = f(x_1, \ldots, x_n)$, where x_i is the input from party p_i, then evaluate $\text{STATE}_{\text{C}}^{n-1}$ on y, and finally output to each party its corresponding share, resulting in $[y]$. If GMW aborts with a set B of active cheaters, repeat with $\mathcal{P} = \mathcal{P} \backslash B$, $n = n - |B|$, and $e = e - |B|$.
2. Invoke $\text{REC}_{\text{C}}^{n-1}$ on $[y]$. On abort with a set B of active cheaters: If $|B| \leq e$, then repeat the whole protocol with $\mathcal{P} = \mathcal{P} \setminus B$, $n = n - |B|$, and $e = e - |B|$. Otherwise, halt the execution.
3. Output y.

Figure 6.6: The computationally secure protocol for non-reactive MPC for threshold adversaries.

Lemma 42. *Given a function f and a robustness parameter e, the protocol for non-reactive MPC computes f in presence of an adversary corrupting $(|\mathcal{D}^*|, |\mathcal{E}^*|)$. The protocol always guarantees correctness and agreement on abort. Furthermore, it is robust if $|\mathcal{D}^*| \leq e$, secret if $|\mathcal{D}^*| + |\mathcal{E}^*| < n$ or $|\mathcal{E}^*| < n - e$, and fair if $|\mathcal{D}^*| + |\mathcal{E}^*| < n$.*

Proof. CORRECTNESS follows trivially by inspection.

ROBUSTNESS: Since $|\mathcal{D}^*| \leq e$ and $B \subseteq \mathcal{D}^*$, the protocol is repeated (with $\mathcal{P} = \mathcal{P} \setminus B$) until it succeeds. Since B is non-empty, the protocol is repeated at most n times.

SECRECY: Since $|\mathcal{D}^*| + |\mathcal{E}^*| < n$ or $|\mathcal{E}^*| < n - e$, we have in particular that $|\mathcal{E}^*| \leq n - 1$. Hence, both GMW and the output $[y]$ (Corollary 4) are secret, and the adversary obtains no information about the inputs, outputs, or intermediate results in Step 1. Given the output, Steps 2 and 3 are independent from the inputs and intermediate results. Therefore, if the protocol does not abort, secrecy is maintained. Yet, secrecy may be violated if the adversary can force a repetition of the protocol after learning the output.[29]

[29] In that case, the adversary may learn two evaluations of f for different inputs.

If the protocol aborts in Step 2 with (B, t), we have that $|B| + t + 1 \geq n$. On the one hand, if $|\mathcal{D}^*| + |\mathcal{E}^*| < n$, then since $B \subseteq \mathcal{D}^*$ we directly have $t + 1 \geq n - |B| \geq n - |\mathcal{D}^*| > |\mathcal{E}^*|$, i.e., $|\mathcal{E}^*| \leq t$ and secrecy is maintained. On the other hand, if $|\mathcal{E}^*| < n - e$, then we make a further distinction: If $|\mathcal{E}^*| \leq t$, then again secrecy is maintained. Otherwise, if $|\mathcal{E}^*| \geq t + 1$, we obtain $|B| \geq n - (t+1) \geq n - |\mathcal{E}^*| > e$ and the protocol aborts, i.e. the adversary learns at most one output value.

The proof of FAIRNESS is along the lines of the proof of secrecy, and is therefore omitted. Finally, since the abort decision is based only on broadcasted values, we always have AGREEMENT ON ABORT (cf. Section 2.3.2). □

Given Lemma 42, we can derive a tight bound for non-reactive MPC with threshold adversaries:

Theorem 6. *In the secure channels model with broadcast and multi-threshold adversaries, computationally secure non-reactive MPC among $n \geq 2$ parties with 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*

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

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

Proof. To prove *sufficiency*, we first note that if $T^s = \{(0,0)\}$, there is no secrecy requirement, and we can directly use the trivial non-secret protocol described in Section 4.1.2. Otherwise, we use the protocol described in Figure 6.6 with $e = \hat{t_a^r}$, where $\hat{t_a^r}$ is the maximal t_a^r value in T^r.

CORRECTNESS and AGREEMENT ON ABORT are always guaranteed, and ROBUSTNESS follows directly from the choice of e.

SECRECY: Since $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq T^s$, we immediately have that
$$|\mathcal{D}^*| + |\mathcal{E}^*| < n \ \vee \ \forall (t_a^r, \cdot) \leq T^r : t_a^r + |\mathcal{E}^*| < n.$$
Then, it follows from the choice of e that $|\mathcal{D}^*| + |\mathcal{E}^*| < n \ \vee \ e + |\mathcal{E}^*| < n$.

FAIRNESS: Since $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq T^f$ and $\forall (t_a^f, t_p^f) \leq T^f : t_a^f + t_p^f < n$, we immediately have $|\mathcal{D}^*| + |\mathcal{E}^*| < n$.

The proof of *necessity* follows directly from the corresponding proof for general adversaries. □

6.4 Reactive Multi-Party Computation

For our protocols for reactive MPC, we adapt an idea from [IKLP06] and modify the given functionality $\mathcal{F}$ as follows: For each output y, instead of the value itself, it outputs a sharing of y that was computed according to some VSS scheme. Then, to obtain the output, the parties reconstruct this sharing.

In contrast to [IKLP06], we use a gradual VSS scheme for the modification of $\mathcal{F}$. The gradual property allows to provide fairness beyond robustness. In fact, we only require the (probabilistic) function STATE that, given a value y, samples shares of y according to the gradual VSS scheme. We modify $\mathcal{F}$ such that it invokes STATE on each output value y, and then outputs the shares of y (instead of y itself). When we modify $\mathcal{F}$ using $\text{VSS}_{\text{C}}^{\mathcal{S}}$ (Section 6.1.2), we denote the resulting functionality by $\mathcal{F}^{\mathcal{S}}$, and when using $\text{VSS}_{\text{C}}^{d}$ (Section 6.1.3) with $\mathcal{F}^{d}$.

To implement the (modified) functionality $\mathcal{F}$, we use the MPC protocols described in Section 6.2.2.2, i.e., $\texttt{GMW}^{\mathcal{S}}$ for the setting with general adversaries and $\texttt{GMW}^{d}$ for the threshold setting. These protocols receive as parameter a $(\mathcal{Z}^s, \mathcal{Z}^r)$-secure VSS, and then provide correctness for any number of corrupted parties, secrecy if $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^s$, and robustness if $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^r$. Furthermore, if the protocol is aborted, then each party outputs the same non-empty set $B \subseteq \mathcal{D}^*$.

6.4.1 Reactive MPC for General Adversaries

In the setting with general adversaries, we use the gradual VSS scheme introduced in Section 6.1.2 with a sharing specification $\mathcal{S}$ for the modification of $\mathcal{F}$, resulting in $\mathcal{F}^{\mathcal{S}}$, and with a sharing specification $\mathcal{S}'$ within $\texttt{GMW}$, resulting in $\texttt{GMW}^{\mathcal{S}'}$. That means, each output y is shared with an $\mathcal{S}$-sharing, resulting in $[y]^{\mathcal{S}}$.

Lemma 43. *Given a functionality $\mathcal{F}$, and sharing specifications $\mathcal{S}$ and $\mathcal{S}'$ with $\emptyset \notin \mathcal{S}$ and with $\emptyset \notin \mathcal{S}'$, the protocol for reactive MPC implements $\mathcal{F}$ in presence of an adversary corrupting $(\mathcal{D}^*, \mathcal{E}^*)$. The protocol always guarantees correctness and agreement on abort. Furthermore, it is secret if $\exists S_i \in \mathcal{S}' : S_i \cap \mathcal{E}^* = \emptyset$, robust if $\forall S_i \in \mathcal{S}' : S_i \not\subseteq \mathcal{D}^*$ and $\forall S_i \in \mathcal{S} : S_i \not\subseteq \mathcal{D}^*$, and fair if $\exists S_i \in \mathcal{S}' : S_i \cap \mathcal{E}^* = \emptyset$ and*

$$\forall S_i \in \mathcal{S} : \big(S_i \not\subseteq \mathcal{D}^* \ \vee \ \exists S_j \in \mathcal{S}, j > i : S_j \cap \mathcal{E}^* = \emptyset\big).$$

Protocol for reactive MPC: Given is a functionality $\mathcal{F}$.

1. Invoke $\texttt{GMW}^{\mathcal{S}'}$ implementing $\mathcal{F}^{\mathcal{S}}$.
2. On each output $[y]^{\mathcal{S}}$, invoke $\text{REC}_{\text{C}}^{\mathcal{S}}$. If it aborts, halt the execution. Otherwise, output y.

Figure 6.7: The computationally secure protocol for reactive MPC for general adversaries.

Proof. CORRECTNESS follows trivially by inspection, and SECRECY and ROBUSTNESS follow immediately from Corollary 3. Since the abort decision is based only on broadcasted values, we always have AGREEMENT ON ABORT (cf. Section 2.3.2).

FAIRNESS: Since $\exists S_i \in \mathcal{S}' : S_i \cap \mathcal{E}^* = \emptyset$, it follows from Corollary 3 that the adversary obtains no information in Step 1. Furthermore, if the reconstruction of an output value aborts with (B, i), the gradual property guarantees that $B \subseteq \mathcal{D}^*$ and $B \cup L_i = \mathcal{P}$. Since $\mathcal{L} = \overline{\mathcal{S}}$ (Corollary 3), this implies $S_i \subseteq \mathcal{D}^*$. Then, it follows from the second part of the condition in the lemma that $\exists S_j \in \mathcal{S}, j > i : S_j \cap \mathcal{E}^* = \emptyset$. That means that the adversary did not obtain any information about y and fairness is preserved. □

Given Lemma 43, we can derive a tight bound for reactive MPC with general adversaries:

Theorem 7. *In the secure channels model with broadcast and general adversaries, computationally 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*

$$\Big(\ \forall(\cdot, \mathcal{E}^s) \in \mathcal{Z}^s, (\mathcal{D}^r, \cdot) \in \mathcal{Z}^r : \ \mathcal{D}^r \cup \mathcal{E}^s \neq \mathcal{P}$$
$$\wedge \ \exists \text{an ordering of } \mathcal{Z}^f, \forall(\mathcal{D}_k^f, \cdot), (\cdot, \mathcal{E}_i^f) \in \mathcal{Z}^f, k \leq i : \ \mathcal{D}_k^f \cup \mathcal{E}_i^f \neq \mathcal{P} \Big)$$
$$\vee \quad \mathcal{Z}^s = \{(\emptyset, \emptyset)\}$$

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

Proof. To prove *sufficiency*, we first note that if $\mathcal{Z}^s = \{(\emptyset, \emptyset)\}$, there is no secrecy requirement, and we can directly use the trivial non-secret protocol described in Section 4.1.2. Otherwise, we use the protocol described

in Figure 6.7 with $\mathcal{S}' := (\overline{\mathcal{E}^s} \mid (\cdot, \mathcal{E}^s) \in \mathcal{Z}^s)$ (with an arbitrary order), and $\mathcal{S} := \left(\overline{\mathcal{E}_i^f} \mid (\cdot, \mathcal{E}_i^f) \in \mathcal{Z}^f\right)$ (maintaining the order of $\mathcal{Z}^f$). It follows from the condition in the theorem that $\forall(\emptyset, \mathcal{E}^s) \in \mathcal{Z}^s : \mathcal{E}^s \neq \mathcal{P}$, and hence $\emptyset \notin \mathcal{S}'$. Since $\mathcal{Z}^f \subseteq \mathcal{Z}^s$, we also have $\emptyset \notin \mathcal{S}$.

CORRECTNESS and AGREEMENT ON ABORT are always guaranteed, and SECRECY follows immediately from the choice of $\mathcal{S}'$.

ROBUSTNESS: Since $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^r$ and $\forall(\cdot, \mathcal{E}^s) \in \mathcal{Z}^s, (\mathcal{D}^r, \cdot) \in \mathcal{Z}^r : \mathcal{D}^r \cup \mathcal{E}^s \neq \mathcal{P}$ we have that $\forall(\cdot, \mathcal{E}^s) \in \mathcal{Z}^s : \mathcal{D}^* \cup \mathcal{E}^s \neq \mathcal{P}$, i.e. $\forall(\cdot, \mathcal{E}^s) \in \mathcal{Z}^s : \mathcal{P} \setminus \mathcal{E}^s \not\subseteq \mathcal{D}^*$. Then it follows from the choice of $\mathcal{S}'$ that $\forall S_i \in \mathcal{S}' : S_i \not\subseteq \mathcal{D}^*$. Since $\mathcal{Z}^f \subseteq \mathcal{Z}^s$, we immediately also have $\forall S_i \in \mathcal{S} : S_i \not\subseteq \mathcal{D}^*$.

FAIRNESS: Given is that $(\mathcal{D}^*, \mathcal{E}^*) \in \mathcal{Z}^f$. Since $\mathcal{Z}^f \subseteq \mathcal{Z}^s$, we have that $\exists S_i \in \mathcal{S}' : S_i \cap \mathcal{E}^* = \emptyset$. Furthermore, since $\mathcal{Z}^f$ is monotone, there is an index j such that $(\mathcal{D}^*, \mathcal{E}^*) = (\mathcal{D}_j^f, \mathcal{E}_j^f)$. Let i be an arbitrary index in $\mathcal{Z}^f$. If $j \leq i$, then it follows from the condition in the theorem that $\mathcal{P} \setminus \mathcal{E}_i^f \not\subseteq \mathcal{D}^*$, i.e., given the choice of $\mathcal{S}$, that $S_i \not\subseteq \mathcal{D}^*$. Otherwise, if $j > i$, then given the choice of $\mathcal{S}$ we have that $\exists S_j \in \mathcal{S}, j > i : S_j \cap \mathcal{E}^* = \emptyset$.

Now, we prove that the bound in the theorem is also *necessary*, i.e. if violated, (reactive) MPC is impossible. The proof is inspired by [HMZ08]. The bound in the theorem is violated if

$$\Big(\quad \exists(\cdot, \mathcal{E}^s) \in \mathcal{Z}^s, (\mathcal{D}^r, \cdot) \in \mathcal{Z}^r : \mathcal{D}^r \cup \mathcal{E}^s = \mathcal{P} \tag{1}$$

$$\vee \ \forall \text{orderings of } \mathcal{Z}^f, \exists(\mathcal{D}_k^f, \cdot), (\cdot, \mathcal{E}_i^f) \in \mathcal{Z}^f : k \leq i \ \wedge \ \mathcal{D}_k^f \cup \mathcal{E}_i^f = \mathcal{P}\Big) \tag{2}$$

$$\wedge \quad \mathcal{Z}^s \neq \{(\emptyset, \emptyset)\}$$

Case (1). We first consider the case where $\mathcal{Z}^s \neq \{(\emptyset, \emptyset)\}$ and

$$\exists(\cdot, \mathcal{E}^s) \in \mathcal{Z}^s, (\mathcal{D}^r, \cdot) \in \mathcal{Z}^r : \mathcal{D}^r \cup \mathcal{E}^s = \mathcal{P}.$$

Since $\mathcal{Z}^s \neq \{(\emptyset, \emptyset)\}$, we can assume that $\mathcal{E}^s \neq \emptyset$. Furthermore, if $\mathcal{D}^r = \emptyset$, we have $\mathcal{E}^s = \mathcal{P}$, and MPC is impossible as stated in Section 6.2.2.1. Assume there is a protocol for such non-empty $\mathcal{E}^s$ and $\mathcal{D}^r$. Then, the adversary corrupts $(\mathcal{D}^*, \mathcal{E}^*) = (\mathcal{D}^r, \mathcal{E}^r)$ and has the parties in $\mathcal{D}^r$ stop sending messages. Since the remaining parties are covered by $\mathcal{E}^s$, the state is lost and the computation cannot be continued. Hence, robustness is violated.[30]

[30] Note that the proof in [IKLP06] considers only the threshold case, and only the special case where $t_a^r \leq t_p^s$.

Case (2). Now, we consider the case that $\mathcal{Z}^s \neq \{(\emptyset, \emptyset)\}$ and

$$(\forall \text{orderings of } \mathcal{Z}^f, \exists(\mathcal{D}_k^f, \cdot), (\cdot, \mathcal{E}_i^f) \in \mathcal{Z}^f : \; k \leq i \; \wedge \; \mathcal{D}_k^f \cup \mathcal{E}_i^f = \mathcal{P}).$$

We can assume that the sets $\mathcal{D}_k^f$ and $\mathcal{E}_i^f$ are non-empty: If either $\mathcal{D}_k^f = \emptyset$ or $\mathcal{E}_i^f = \emptyset$, we have $\mathcal{E}_i^f = \mathcal{P}$ or $\mathcal{D}_k^f = \mathcal{P}$, respectively. In both cases, we have $\mathcal{E}^s = \mathcal{P}$ for some $\mathcal{E}^s$, and MPC is impossible as stated in Section 6.2.2.1. Assume there is a protocol for such non-empty $\mathcal{D}_k^f$ and $\mathcal{E}_i^f$. Every protocol defines an ordering in which the sets $\mathcal{E}^f$ (for $(\cdot, \mathcal{E}^f) \in \mathcal{Z}^f$) learn the result.[31] The assumption states that for every ordering of $\mathcal{Z}^f$, there are indices $k \leq i$ such that $\mathcal{D}_k^f \cup \mathcal{E}_i^f = \mathcal{P}$. Assume that the parties in $\mathcal{E}_k^f$ learn the result in round ℓ. Now, the adversary corrupts $(\mathcal{D}^*, \mathcal{E}^*) = (\mathcal{D}_k^f, \mathcal{E}_k^f)$, and has the parties in $\mathcal{D}_k^f$ stop sending any messages in and after round ℓ. The remaining parties are covered by $\mathcal{E}_i^f$. Hence, after round $\ell - 1$ they do not yet hold enough information to efficiently reconstruct the result, while the adversary does learn the result. This constitutes a violation of fairness. □

6.4.2 Reactive MPC for Threshold Adversaries

In the setting with threshold adversaries, we use the gradual VSS scheme described in Section 6.1.3 with the same sharing degree d for both the modification of $\mathcal{F}$, resulting in $\mathcal{F}^d$, and within GMW, resulting in GMW^d.

Protocol for reactive MPC: Given is a functionality $\mathcal{F}$.

1. Invoke GMW^d implementing $\mathcal{F}^d$.
2. On each output $[y]$, invoke REC_{C}^d. If it aborts, halt the execution. Otherwise, output y.

Figure 6.8: The computationally secure protocol for reactive MPC for threshold adversaries.

Lemma 44. *Given a functionality $\mathcal{F}$ and a parameter $d < n$, the protocol for reactive MPC implements $\mathcal{F}$ in presence of an adversary corrupting $(|\mathcal{D}^*|, |\mathcal{E}^*|)$. The protocol always guarantees correctness and agreement on abort. Furthermore, it is secret if $|\mathcal{E}^*| \leq d$, robust if $|\mathcal{D}^*| < n - d$, and fair if $|\mathcal{E}^*| \leq d \; \wedge \; |\mathcal{D}^*| + |\mathcal{E}^*| < n$.*

[31] I.e., when the parties in $\mathcal{E}^f$ jointly hold enough information to be able to efficiently reconstruct the result.

Proof. CORRECTNESS follows trivially by inspection, and SECRECY and ROBUSTNESS follow immediately from Corollary 4. Since the abort decision is based only on broadcasted values, we always have AGREEMENT ON ABORT (cf. Section 2.3.2).

FAIRNESS: Since $|\mathcal{E}^*| \leq d$, it follows from Corollary 4 that the adversary obtains no information in Step 1. Furthermore, if the reconstruction of an output value aborts with (B, t), the gradual property guarantees that $|B| + t + 1 \geq n$ and $B \subseteq \mathcal{D}^*$. Since $|\mathcal{D}^*| + |\mathcal{E}^*| < n$, we then have $t + 1 \geq n - |B| \geq n - |\mathcal{D}^*| > |\mathcal{E}^*|$, i.e. $|\mathcal{E}^*| \leq t$. Hence, the adversary did not obtain any information about y and fairness is preserved. □

Given Lemma 44, we can derive a tight bound for reactive MPC with threshold adversaries:

Theorem 8. *In the secure channels model with broadcast and multi-threshold adversaries, computationally secure reactive MPC among $n \geq 2$ parties with 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*

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

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

Proof. To prove *sufficiency*, we first note that if $T^s = \{(0,0)\}$, there is no secrecy requirement, and we can directly use the trivial non-secret protocol described in Section 4.1.2. Otherwise, we use the protocol described in Figure 6.8 with $d = \hat{t_p^s}$, where $\hat{t_p^s}$ is the maximal t_p^s value in T^s. It follows from the condition in the theorem that $\forall(\cdot, t_p^s) \leq T^s : t_p^s < n$, and hence $d < n$.

CORRECTNESS and AGREEMENT ON ABORT are always guaranteed, and SECRECY follows immediately from the choice of d.

ROBUSTNESS: Since $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq T^r$ and $\forall(t_a^r, \cdot) \leq T^r, (\cdot, t_p^s) \leq T^s : t_a^r + t_p^s < n$, we have that $\forall(\cdot, t_p^s) \leq T^s : |\mathcal{D}^*| + t_p^s < n$. Then, it follows from the choice of d that $|\mathcal{D}^*| + d < n$.

FAIRNESS: Given is that $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq T^f$. Since $T^f \leq T^s$, we have that $|\mathcal{E}^*| \leq d$. Furthermore, since $\forall(t_a^f, t_p^f) \leq T^f : t_a^f + t_p^f < n$, we immediately have $|\mathcal{D}^*| + |\mathcal{E}^*| < n$.

The proof of *necessity* follows directly from the corresponding proof for general adversaries. □

6.5 Summary

We have provided computationally secure MPC protocols with graceful degradation of both security and corruptions, together with tight bounds, for both general and threshold adversaries, as well as for both non-reactive and reactive MPC. Our protocols strictly generalize and extend known results from the literature. In particular, we improved over the work in [IKLP06] that combines optimal results from the active and the passive world. Our protocols distinguish not only whether or not active cheating occurs, but provide a dynamic tradeoff between active and passive corruptions. Hence, we achieve "the best of both worlds – and everything in between" with a single protocol.

To illustrate the gain in security guarantees, consider a setting with $n = 10$ parties. The protocol of [IKLP06] achieves full security for either 9 passively and no actively corrupted parties (i.e., against $(0, 9)$), or against 4 actively (and no additional passively) corrupted parties (i.e., against $(4, 4)$). Our protocol additionally guarantees full security against all of $(4, 5)$, $(3, 6)$, $(2, 7)$, and $(1, 8)$.

Furthermore, we introduced the notion of *gradual* verifiable secret sharing. This notion requires that, during reconstruction, secrecy is given up gradually, one subset at a time, while immediately establishing robustness against the corresponding complement set. As a consequence, intuitively speaking, the adversary might still abort the protocol, but does not automatically learn the secret. This technique turned out to be very useful in the setting of both non-reactive and reactive MPC to provide more flexible and therefore more practical protocols.

Chapter 7

Conclusions

We have provided the first MPC protocols with graceful degradation in multiple dimensions, namely graceful degradation of security, as well as graceful degradation with respect to the corruption type. This covers all common security notions for MPC (correctness, secrecy, robustness, fairness, and agreement on abort), as well as the most prominent corruption types (honest, passive, active). Furthermore, we consider both the model with general and the model with threshold adversaries, for all three security levels (perfect, statistical, and computational). The protocols are strict generalizations (and combinations) of hybrid-secure MPC and mixed adversaries. We derived tight bounds for the existence of secure MPC protocols for the given settings, and have shown that our protocols achieve these bounds.

For each security level (perfect, statistical, or computational), our protocols provide fundamental insights in the field of MPC. In the perfect setting, we have shown that there is a trade-off between the different security properties (correctness, secrecy, robustness, and fairness). Our bounds quantify this trade-off, and point out how much resilience one has to give up on one property, to achieve higher resilience for another property (the same observation also holds for the statistical and the computational settings). Our results in the statistical setting prove that, in a setting with mixed adversaries, passively corrupted parties do not only have the evident impact on secrecy, but in particular also on correctness. This is in contrast to both the perfect and the computational settings. In the computational setting, we find a dynamic trade-off between active and passive corruptions in the threshold setting: Our protocols are secure against several incomparable adversaries simultaneously. In

other words, the protocol does not depend on an assumption how many parties the adversary corrupts actively and passively during the protocol execution, as long as the sum of the total number of corruptions plus the number of active corruptions is less than the total number of parties.

To obtain a high degree of flexibility in our protocols, we parametrize the constructions. This use of parameters also highlights how the different security properties are achieved on a technical level, which allows for a better understanding of why the protocols are the way they are.

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 protocols 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.

Moreover, the use of multi-thresholds allows to unify two incomparable models for combining active and passive corruption. In the first model, used for example by [IKLP06], the adversary can corrupt parties either passively or actively, but not both at the same time. Then, for each of the two corruption options, a maximally tolerable adversary is considered. In the second model, used for example by [FHM98], the adversary can corrupt some parties actively, and additionally some parties passively, at the same time. Yet, their model only allows to consider a single maximally tolerable adversary. By using multi-thresholds, we can provide a single protocol that subsumes results for both models simultaneously.

We leave as open problems to consider additional security properties and corruption types (e.g. fail-corruption), as well as to combine additional dimensions of graceful degradation (like, e.g., efficiency) with graceful degradation of security and corruption types. Furthermore, for the perfect and the statistical setting, we focus on reactive MPC. The bounds for non-reactive MPC might be slightly weaker.

Also, in all our protocols, we assume that parties generate their randomness on the fly ("instant randomness"), which allows even passively corrupted parties to e.g. choose random challenges that are unpredictable to the adversary. It seems clear that, in the perfect setting, a fixed random tape does not affect the security of the protocols. We leave as an open problem to find tight bounds without this assumption for both the statistical and the computational setting.

Bibliography

[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.

[BGP89] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Towards optimal distributed consensus (extended abstract). In *FOCS '89*, pages 410–415. IEEE, 1989.

[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 multiparty computation with dispute control. In *TCC 2006*, pages 305–328. Springer, 2006.

[Blu83] Manuel Blum. How to exchange (secret) keys (extended abstract). In *STOC '83*, pages 440–447. ACM, 1983.

[BP04] Mihir Bellare and Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. In *ASIACRYPT 2004*, pages 48–62. Springer, 2004.

[BPW04] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A general composition theorem for secure reactive systems. In *TCC 2004*, pages 336–354. Springer, 2004.

[BR94] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *EUROCRYPT '94*, pages 92–111. Springer, 1994.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[Can01] Ran Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *FOCS '01*, pages 136–145. IEEE, 2001.

[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.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC '02*, pages 494–503. ACM, 2002.

[CW89] Brian A. Coan and Jennifer L. Welch. Modular construction of nearly optimal Byzantine agreement protocols. In *PODC '89*, pages 295–305. ACM, 1989.

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

[Den06] Alexander W. Dent. The cramer-shoup encryption scheme is plaintext aware in the standard model. In *EUROCRYPT 2006*, pages 289–307. Springer, 2006.

[DS82] Danny Dolev and H. Raymond Strong. Polynomial algorithms for multiple processor agreement. In *STOC '82*, pages 401–407. ACM, 1982.

[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.

[FM98] Matthias Fitzi and Ueli Maurer. Efficient Byzantine agreement secure against general adversaries. In *DISC '98*, pages 134–148. Springer, 1998.

[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.

[Gol04] Oded Goldreich. *Foundations of Cryptography*, volume Basic Applications. Cambridge University Press, 2004.

[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. 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.

[KLR06] Eyal Kushilevitz, Yehuda Lindell, and Tal Rabin. Information-theoretically secure protocols and security under composition. In *STOC '06*, pages 109–118. ACM, 2006.

[Lam83] Leslie Lamport. The weak Byzantine generals problem. *Journal of the ACM*, 30(3):668–676, 1983.

[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.

[LSP82] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

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

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '91*, pages 129–140. Springer, 1991.

[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.