Graceful Degradation in Multi-Party Computation*

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

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

Abstract. The goal of *Multi-Party Computation* (MPC) is to perform an arbitrary computation in a distributed, private, and fault-tolerant way. For this purpose, a fixed set of n parties runs a protocol that tolerates an adversary corrupting a subset of the participating parties, and still preserves certain security guarantees.

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

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

Keywords: Multi-party computation, graceful degradation, hybrid security, mixed adversaries.

1 Introduction

1.1 Secure Multi-Party Computation

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

MPC was originally proposed by Yao [Yao82]. The first general solution was provided in [GMW87], where, based on computational intractability assumptions, security against a passive adversary was achieved for $t < n$ corruptions, and security against an active adversary was achieved for $t < \frac{n}{2}$ corruptions. In [BGW88,CCD88], information-theoretic security 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 results were generalized to the non-threshold setting, where the corruption capability of the adversary is not

* An extended abstract of this paper appeared at ICITS 11 [HLMR11]. This work was partially supported by the Zurich Information Security Center.

specified by a threshold t, but rather by a so-called adversary structure $\mathcal{Z}$, a monotone collection of subsets of the player set, where the adversary can corrupt the players in one of these subsets [HM97].

All mentioned protocols achieve full security, i.e., secrecy, correctness, and robustness. *Secrecy* means that the adversary learns nothing about the honest parties' inputs and outputs (except, of course, for what she can derive from the corrupted parties' inputs and outputs). *Correctness* means that all parties either output the right value or no value at all. *Robustness* means that the adversary cannot prevent the honest parties from learning their respective outputs. This last requirement turns out to be very strong. Therefore, relaxations of full security have been proposed, where robustness is replaced by weaker output guarantees: *Fairness* means that the adversary can possibly prevent the honest parties from learning their outputs, but then also the corrupted parties do not learn their outputs. *Agreement on abort* means that the adversary can possibly prevent honest parties from learning their output, even while corrupted parties learn their outputs, but then the honest parties at least reach agreement on this fact (and typically make no output). Note that for example [GMW87] achieves secrecy, correctness, and agreement on abort (but neither robustness nor fairness) for up to $t < n$ active corruptions.

1.2 Graceful Degradation

Most MPC protocols in the literature do not degrade very gracefully. They provide a very high level of security up to some threshold t, but no security at all beyond this threshold. There are no intermediate levels of security.[3] Furthermore, a party is considered either fully honest or fully corrupted. There are no intermediate levels of corruptions.

Note that 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.

Graceful degradation was first considered by Chaum [Cha89]: He proposed one protocol with graceful degradation of security, namely from information-theoretic security (few corruptions) over computational security (more corruptions) to no security (many corruptions), and another, independent protocol with graceful degradation of corruptions, namely by considering fully honest, passively corrupted, and actively corrupted parties in the same protocol execution. The former protocol (graceful degradation of security, often called *hybrid* security) was recently generalized in [FHHW03,FHW04,IKLP06,Kat07,LRM10]. The latter protocol (graceful degradation of corruptions, often called *mixed* security) was generalized and extended in [DDWY93,FHM98,FHM99,BFH$^+$08,HMZ08].

1.3 Our Focus

In this work, we consider simultaneously graceful degradation of security (i.e., hybrid security) and graceful degradation of corruptions (i.e., mixed adversaries), both in the threshold and in the general adversary setting. 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).[4] We assume that $t^s \leq t^c$ and $t^r \leq t^c$, since secrecy and robustness are not well defined in a setting without correctness. Furthermore, we assume that $t^f \leq t^s$ since in a setting without secrecy the adversary inherently has an unfair advantage over honest parties.

[3] The same observation holds for known protocols for general adversaries.

[4] If the number of corruptions is below multiple thresholds, all corresponding security properties are achieved. In particular, full security is achieved if the number of corruptions is below all thresholds.

Furthermore, we also consider graceful degradation with respect to the corruption type: We consider, at the same time, honest parties, passively corrupted parties, and actively corrupted parties (so-called *mixed adversaries*). Such an adversary 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. Note 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, $t_a \leq t_p$).

In the non-threshold setting, security is characterized by four adversary structures $\mathcal{Z}^c$, $\mathcal{Z}^s$, $\mathcal{Z}^r$, $\mathcal{Z}^f$, where correctness, secrecy, robustness, and fairness are guaranteed as long as the set of corrupted players is contained in the corresponding adversary structure.[5] As argued above, we assume that $\mathcal{Z}^s \subseteq \mathcal{Z}^c$, $\mathcal{Z}^r \subseteq \mathcal{Z}^c$ and $\mathcal{Z}^f \subseteq \mathcal{Z}^s$. In order to model both passive and active corruptions, each adversary structure consists of tuples $(\mathcal{D}, \mathcal{E})$ of subsets of the player set, where $\mathcal{E}$ is the set of passively (eavesdropping), and $\mathcal{D} \subseteq \mathcal{E}$ is the set of actively (disruption) corrupted parties. A protocol with adversary structure $\mathcal{Z}$ provides security guarantees for every adversary actively corrupting the parties in $\mathcal{D}$ and passively corrupting the parties in $\mathcal{E}$, for some $(\mathcal{D}, \mathcal{E}) \in \mathcal{Z}$.

Note that the notion of correctness for a security level without secrecy differs from the usual interpretation: The adversary is rushing and may know the entire state of the protocol execution. Hence, input-independence cannot be achieved. Furthermore, for the same reason, we can have probabilistic computations only with adversarially chosen randomness.

1.4 Contributions

We provide the first MPC protocol with graceful degradation in multiple dimensions: We consider all security properties generally discussed in the literature (secrecy, correctness, robustness, fairness, and agreement on abort), and the most prominent corruption types (active, passive). We prove a tight bound on the feasibility of perfectly-secure MPC, both in the threshold and the non-threshold setting, and provide efficient perfectly-secure general MPC protocols matching these bounds.[6] Our main results (Theorems 1 and 2) are a strict generalization of the previous results for perfect MPC, which appear as special cases in our unified treatment. For the sake of simplicity, we do not include fail corruption [BFH$^+$08]. Note that fairness is not discussed in the protocol descriptions, but in Section 4.

Previous results for perfectly secure MPC considered graceful degradation only of corruption levels, i.e., the known protocols always provide full security. 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 separates the different security properties, and therefore allows to make precise statements formalizing the above intuition. This indicates that our model is both natural and appropriate.

1.5 Motivating Example

The strength of our result is the possibility to provide protocols that are tailored much more precisely to the security requirements of a specific setting than previous solutions: As a simple example consider voting. A solution based on a traditional perfectly secure MPC protocol, e.g. [BGW88], achieves secrecy and correctness for up to $t < \frac{n}{3}$ corrupted parties, but provides

[5] As in the threshold case, if the set of corrupted parties is contained in multiple adversary structures, all corresponding security properties are achieved.

[6] The protocols are efficient in the input length, i.e. the threshold protocol is efficient in the number of parties and the size of the circuit to be computed, whereas the protocol for general adversaries is efficient in the size of the adversary structure and the size of the circuit.

no guarantees if $t \geq \frac{n}{3}$. However, in voting it is generally much more important that the final tally is correct than to protect the secrecy of votes. Our protocol allows to reduce secrecy to $t = \frac{n}{8}$ corrupted parties, while guaranteeing correctness for $t < \frac{3n}{4}$ actively corrupted parties (and additionally arbitrarily many passively corrupted parties). This protocol is robust for up to $t = \frac{n}{8}$ corruptions. It is also possible to trade correctness for robustness: By reducing the correctness guarantee to $t < \frac{n}{2}$ corruptions, robustness is guaranteed for up to $t = \frac{3n}{8}$ corruptions.

1.6 Model

We consider n parties $1, \ldots, n$, connected by pairwise synchronous secure channels, who want to compute some probabilistic function over a finite field $\mathbb{F}$, represented as a circuit with input, addition, multiplication, random, and output gates. This function can be reactive, where parties can provide further inputs after having received some intermediate outputs. In the main body of this paper, we assume that authenticated broadcast channels are given. The model without broadcast channels is treated in Appendix B.

There is a central adversary with unlimited computing power who corrupts some parties passively (and reads their internal state) or even actively (and makes them misbehave arbitrarily). We denote the actual sets of actively (passively) corrupted parties by $\mathcal{D}^*$ ($\mathcal{E}^*$), where $\mathcal{D}^* \subseteq \mathcal{E}^*$. Uncorrupted parties are called *honest*, non-active parties are called *correct*. The security of our protocols is perfect, i.e., information-theoretic with no error probability. The level of security (secrecy, correctness, fairness, robustness, agreement on abort) depends on $(\mathcal{D}^*, \mathcal{E}^*)$.

For ease of notation, we assume that if a party does not receive an expected message (or receives an invalid message), a default message is used instead.

1.7 Outline of the Paper

Our paper is organized as follows: As a main technical contribution, we generalize known protocols for threshold and general adversaries in Sections 2 and 3. In Section 4, we state optimal bounds for MPC, together with proofs of sufficiency. Tightness of the bounds is proven in Section 5.

2 A Parametrized Protocol for Threshold Adversaries

In this section, we generalize the perfectly secure MPC protocol of [BGW88] by introducing two parameters. 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. In case of [BGW88], this is the degree d of the sharing polynomial (see also [FHM98]). Second, given the parameter d for secrecy, we express the reconstruct protocol in terms of an additional parameter determining the amount of error correction taking place. 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. In case of [BGW88], this parameter is the number e of corrected errors during reconstruction. To our knowledge, such a second parameter has not been considered before. The two parameters must fulfill $d + 2e < n$. Note that by 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 protocols and analyze them with respect to correctness, secrecy, and robustness. Note that fairness is discussed in Section 4.

2.1 The Underlying Verifiable Secret Sharing

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

Definition 1 (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 party i holds a share $s_i = \hat{s}(\alpha_i)$. We denote a d-sharing of s with $[s]$, and the share s_i with $[s]_i$. A sharing degree d is* t-permissive *if the shares of all but t parties uniquely define the secret, i.e., $n - t > d$.*

Lemma 1. *Let $d < n$ be the sharing degree. A d-sharing is secret if $|\mathcal{E}^*| \leq d$, and uniquely defines a value if d is $|\mathcal{D}^*|$-permissive.*

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

The share protocol takes as input a secret s from a dealer, and outputs a d-sharing $[s]$ (see Figure 1).

SHARE: Given input s from the dealer, SHARE computes 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 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 (i, j), party i sends $h_i(\alpha_j)$ to party j, and party 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 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 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 i outputs $s_i := k_i(0)$, and the dealer outputs $\hat{s}(x) := g(x, 0)$.[7]

Fig. 1. The Share Protocol.

Lemma 2. *Let $d < n$ be the sharing degree. On input s from the dealer,* SHARE *correctly, secretly, and robustly computes a d-sharing. If d is $|\mathcal{D}^*|$-permissive, and if the dealer is correct, the sharing uniquely defines the secret s.*

Proof. Secrecy: 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

[7] 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.

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. □

The public reconstruction 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 $\frac{n-d-1}{2}$ errors (for robustness). In our protocol, we trade correctness for 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 2) 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 Hamming distance between the shares and the decoded codeword is larger than e. Note that during public reconstruction, there is no secrecy requirement.

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

1. Each party i broadcasts its share s_i. Let $\boldsymbol{s} = (s_1, ..., s_n)$ denote the vector of broadcasted shares.
2. Each party identifies the closest codeword $\boldsymbol{s}_c$ (e.g. using the Berlekamp-Welch algorithm). If the Hamming distance between $\boldsymbol{s}_c$ and $\boldsymbol{s}$ is larger than e, the protocol is aborted. Otherwise, each party interpolates the entries in $\boldsymbol{s}_c$ with a polynomial $\hat{s}_c(x)$ of degree d, and outputs $\hat{s}_c(0)$.[8]

Fig. 2. The Public Reconstruction Protocol.

Lemma 3. *Let d be the sharing degree, and e be the correction parameter, where $d + 2e < n$. Given a d-sharing $[s]$ of some value s,* PUBLIC RECONSTRUCTION *is correct if $|\mathcal{D}^*| < (n - d) - e$, is robust if $|\mathcal{D}^*| \leq e$, and always guarantees agreement on abort.*

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

Agreement on abort: The abort decision is only based on broadcasted values. Hence, either all correct parties abort, or all correct parties continue. □

During PUBLIC RECONSTRUCTION, all parties learn the value under consideration. PRIVATE RECONSTRUCTION, where a value s is disclosed only to a single party k, can be reduced to PUBLIC RECONSTRUCTION using a simple blinding technique ([CDG88]): Party k first shares a uniform random value, which is added to s before PUBLIC RECONSTRUCTION is invoked. Hence, PRIVATE RECONSTRUCTION provides the same security guarantees as PUBLIC RECONSTRUCTION, and additionally provides secrecy of the reconstructed value. Note that the trivial solution, where each party sends its share to party k, does not achieve agreement on abort.

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

2.2 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 i share a random value r_i, and computing $[r] = [r_1] + \ldots + [r_n]$.

The multiplication protocol 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 $v_i = a_i b_i$. This proof requires that a_i and b_i are d-shared, which we achieve by upgrading 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 i knows the sharing polynomial $g(x)$ corresponding to $[z]^{2d}$. However, party i cannot simply broadcast $g(x)$, 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 i broadcasted the correct polynomial $g(x)$, i.e. for party j whether $g(\alpha_j) = [z]_j^{2d}$. Two polynomials of degree $2d$ are equal if they coincide in $2d + 1$ points. So, if party i broadcasts an incorrect $g(x)$, 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.

MULTIPLICATION : Given $[a]$ and $[b]$, the multiplication protocol computes $[c]$ for $c = ab$.

1. Each party i computes $v_i = a_i b_i$, and invokes SHARE on v_i, resulting in $[v_i]$.
2. Invoke UPGRADE 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 on $[a_i]$, $[b_i]$, and $[v_i]$. If the proof is rejected, invoke PUBLIC RECONSTRUCTION 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[9] on $[v_1], \ldots, [v_n]$ for $c = v(0)$, and output the resulting $[c]$.

Fig. 3. The Multiplication Protocol.

Lemma 4. *Let d be the sharing degree, and e be the correction parameter, where $d + 2e < n$. Given d-sharings of a and b,* MULTIPLICATION *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 protocol (Figure 4): Given a d-sharing $[s]$ for some value s, the UPGRADE protocol computes d-sharings $[s_i]$ of all shares s_i.

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

UPGRADE: Given $[s]$, UPGRADE computes $[s_i]$ 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 i chooses a uniformly random value $r^{(i)}$, and invokes SHARE on $r^{(i)}$. By keeping the second dimension at the end of SHARE, 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 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 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.

Fig. 4. Upgrading a d-sharing.

Lemma 5. *Let d be the sharing degree, and e be the correction parameter, where $d + 2e < n$, and assume that* SHARE *and* PUBLIC RECONSTRUCTION *are secure. Given a d-sharing of some value s,* UPGRADE *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 and PUBLIC RECONSTRUCTION: Let $\hat{g}(x, y)$ 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 $\hat{h}(x, y) = \hat{g}(x, y) - \hat{q}(x) = \hat{g}(x, y) - (\hat{g}(x, 0) - \hat{s}(x))$. Hence, $\hat{h}(x, 0) = \hat{s}(x)$ (and $s'_i = s_i$ and $s' = s$) and $h(\alpha_i, y)$ is a random[10] polynomial of degree d, as required. Note that secrecy is guaranteed in the sense that the protocol does not leak more information than the specified output.

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 5).

ABC-PROOF : Given $[a_i]$, $[b_i]$, and $[v_i]$ that are known to party i, ABC-PROOF checks whether $v_i = a_i b_i$.

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

Fig. 5. A Protocol for proving that $c = ab$.

Lemma 6. *Let d be the sharing degree, and e be the correction parameter, where $d + 2e < n$, and assume that* SHARE *and* PUBLIC RECONSTRUCTION *are secure. Given d-sharings $[a_i]$, $[b_i]$, and $[v_i]$,*

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

ABC-PROOF *is correct if* $|\mathcal{D}^*| < n - 2d$, *is secret given that the subprotocols are correct, and is always robust.*

Proof. Secrecy: If party i is correct, then $[0]^{2d}$ (Step 1) is a uniformly random $2d$-sharing of the value 0 [BGW88]. Hence, the same holds for the sharing $[z]^{2d}$ computed in Step 2. If in Step 5 the shares of a party j are reconstructed, either party i or party 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, that $g(x)$ is a polynomial of degree $2d$, and that $[0]^{2d}$ is a $2d$-sharing containing the value 0 (by construction), Now, if $g(x) = 0$ and $g(x)$ is the sharing polynomial corresponding to $[z]^{2d} = [0]^{2d} + [a_i][b_i] - [v_i]$, then $v_i = a_i b_i$. Therefore, we only need to verify that the broadcasted polynomial $g(x)$ is correct.

In Step 3, the sharing polynomial of $[z]^{2d}$ is fixed. Assume that, in this step, party i broadcasts a wrong polynomial $g'(x)$.[11] Note that $g'(x)$ 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 j it holds that $g(\alpha_j) = [z]_j^{2d}$, the polynomials must be equal.

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 and *Agreement on abort:* Both security requirements depend only on the subprotocols.

2.3 The Security of the Parametrized Protocol

Considering the security of the protocols described above, we can derive the security of the parametrized protocol, denoted by $\pi^{d,e}$ (proof ommitted):

Lemma 7. *Let d be the sharing degree, and e be the correction parameter, where $d + 2e < n$. Protocol $\pi^{d,e}$ 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.*

3 A Parametrized Protocol for General Adversaries

For general adversaries, we proceed along the lines of the threshold case: We generalize the protocol of [Mau02] and introduce the *sharing specification* $\mathcal{S} = (S_1, \ldots, S_k)$ (corresponding to the sharing degree d), and the *correction structure* $\mathcal{C} = \{C_1, \ldots, C_l\}$ (corresponding to the correction parameter e), both collections of subsets of $\mathcal{P}$.

3.1 The Underlying Verifiable Secret Sharing

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

Definition 2 ($\mathcal{S}$-Sharing). *A value s is $\mathcal{S}$-*shared *for sharing specification $\mathcal{S} = (S_1, \ldots, S_k)$ if there are values $s_1, \ldots, s_k$, such that $s_1 + \ldots + s_k = s$ and, for all i, every (correct) party $j \in S_i$ holds the summand s_i. A sharing specification $\mathcal{S}$ is $\mathcal{D}$-*permissive, *if each summand is held by at least one party outside $\mathcal{D}$, i.e. $\forall i : S_i \setminus \mathcal{D} \neq \emptyset$.*

Lemma 8. *Let $\mathcal{S}$ be the sharing specification. An $\mathcal{S}$-sharing is secret if $\exists S_i \in \mathcal{S} : S_i \cap \mathcal{E}^* = \emptyset$, and uniquely defines a value if $\mathcal{S}$ is $\mathcal{D}^*$-permissive.*

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

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

The share protocol takes as input a secret s from a dealer, and outputs an $\mathcal{S}$-sharing of the secret s (see Figure 6).

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

1. Let $k = |\mathcal{S}|$. The dealer chooses uniformly random summands $s_1, \ldots, s_{k-1}$ and computes $s_k = s + \sum_{i=1}^{k-1} s_i$. Then, the dealer sends s_i to every party $j \in S_i$.
2. For all $S_i \in \mathcal{S}$: Every party $j \in S_i$ 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. If the dealer does not broadcast a summand s_i, the parties use $s_i = 0$.
4. Each party j outputs its share $\{s_i \mid j \in S_i\}$.

Fig. 6. The Share Protocol for General Adversaries.

Lemma 9. *Let $\mathcal{S}$ be the sharing specification. On input s from the dealer,* SHARE$^{\mathrm{GA}}$ *correctly, secretly and robustly computes an $\mathcal{S}$-sharing. If $\mathcal{S}$ is $\mathcal{D}^*$-permissive, and if the dealer is correct, the sharing uniquely defines the secret s.*

Proof. Secrecy: Given a correct dealer, 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 j with $j \in S_i$ 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[12] of a shared value, we modify the reconstruction protocol of [Mau02]. In our protocol, we trade correctness for robustness by introducing a correction structure $\mathcal{C}$. First, each summand s_i is broadcasted by all parties in S_i. Then, if the inconsistencies can be explained with a faulty set $C \in \mathcal{C}$, the values from parties in C are ignored (corrected), and reconstruction proceeds. Otherwise, the protocol is aborted.

Note that, whenever two sets of possibly actively corrupted parties cover a set $S_i \in \mathcal{S}$, i.e. $S_i \subseteq \mathcal{D}_1 \cup \mathcal{D}_2$, 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 $\mathcal{C}$, namely $\forall S \in \mathcal{S}, C_1, C_2 \in \mathcal{C} : S \not\subseteq C_1 \cup C_2$. However, instead of always correcting as many errors as possible, the protocol allows to select a structure $\mathcal{C}$ that remains below this upper bound (i.e. contains smaller sets C). Now, when correcting errors in a set $C \in \mathcal{C}$, we can detect errors in sets $\mathcal{D}$ where $\forall S_i \in \mathcal{S}, C \in \mathcal{C} : S_i \not\subseteq \mathcal{D} \cup C$. Hence, this approach provides a tradeoff between reduced robustness and extended correctness.

Lemma 10. *Let $\mathcal{S}$ be the sharing specification, and $\mathcal{C}$ be the correction structure, where $\forall S \in \mathcal{S}, C_1, C_2 \in \mathcal{C} : S \not\subseteq C_1 \cup C_2$. Given an $\mathcal{S}$-sharing of some value s,* PUBLIC RECONSTRUCTION$^{\mathrm{GA}}$ *is correct if $\forall C \in \mathcal{C}, S \in \mathcal{S} : S \setminus C \not\subseteq \mathcal{D}^*$, is robust if $\mathcal{D}^* \in \mathcal{C}$, and always guarantees agreement on abort.*

[12] The reduction of private to public reconstruction can be done along the lines of the threshold case.

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

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

Fig. 7. The Public Reconstruction Protocol for General Adversaries.

Proof. Correctness: The condition $\forall C \in \mathcal{C}, S \in \mathcal{S} : S \setminus C \not\subseteq \mathcal{D}^*$ states that for every summand s_i and every set $C \in \mathcal{C}$, 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. If $\mathcal{D}^* \in \mathcal{C}$, these inconsistencies can be explained with a set in $\mathcal{C}$. Hence, the corresponding set can be ignored and reconstruction terminates without abort.

Agreement on abort: The abort decision is based only on broadcasted values. Hence, either all correct parties abort, or all correct parties continue. □

3.2 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 i share a random value r_i, and computing a sharing of $r = r_1 + \ldots + r_n$.

For the multiplication of two values a and b, we use the protocol from [Mau02], based on our modified share and reconstruct protocols. The multiplication protocol exploits the fact that $ab = \sum_{i=1}^{k} \sum_{j=1}^{k} a_i b_j$: 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 choose a correct 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 party is actively corrupted, and the summands a_i and b_j can be reconstructed without violating secrecy.

MULTIPLICATION$^{\text{GA}}$: Given $\mathcal{S}$-sharings of some values $a = \sum_{i=1}^{k} a_i$ and $b = \sum_{i=1}^{k} b_i$, MULTIPLICATION computes 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$^{\text{GA}}$ 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$^{\text{GA}}$ 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$^{\text{GA}}$, 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 $c = ab$.

Fig. 8. The Multiplication Protocol for General Adversaries.

Lemma 11. *Let $\mathcal{S}$ be the sharing and $\mathcal{C}$ be the correction structure, where $\forall S \in \mathcal{S}, C_1, C_2 \in \mathcal{C} : S \not\subseteq C_1 \cup C_2$. 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,*[13] *is secret if the subprotocols are secret and correct, and is robust if the subprotocols are robust.*

Proof. Secrecy: To break secrecy, the adversary has to provoke the reconstruction of summands a_i and b_j where $(S_i \cap S_j) \cap \mathcal{D}^* = \emptyset$, i.e. where the adversary either did not know a_i or did not know b_j beforehand. Since $(S_i \cap S_j) \cap \mathcal{D}^* = \emptyset$, 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 party in $S_i \cap S_j$ that correctly computes and shares this value, which is guaranteed by the premise in the lemma.

Robustness and *Agreement on abort:* Both security requirements depend only on PUBLIC RECONSTRUCTION$^{\text{GA}}$ and SHARE$^{\text{GA}}$.

3.3 The Security of the Generalized Protocol from [Mau02]

Considering the security of the protocols described above, we can derive the security of the protocol described above, denoted by $\pi^{\mathcal{S},\mathcal{C}}$ (proof ommitted):

Lemma 12. *Let $\mathcal{S}$ be the sharing specification, and $\mathcal{C}$ be the correction structure, where $\forall S \in \mathcal{S}, C_1, C_2 \in \mathcal{C} : S \not\subseteq C_1 \cup C_2$. The protocol $\pi^{\mathcal{S},\mathcal{C}}$ guarantees* correctness *if* $\forall S_i, S_j \in \mathcal{S} : S_i \cap S_j \not\subseteq \mathcal{D}^*$ and $\forall C \in \mathcal{C}, S \in \mathcal{S} : S \setminus C \not\subseteq \mathcal{D}^*$, secrecy *if $\exists S_i \in \mathcal{S} : S_i \cap \mathcal{E}^* = \emptyset$ and correctness is guaranteed,* robustness *if $\mathcal{D}^* \in \mathcal{C}$, and* agreement on abort *always.*

4 The Main Results

The following theorems state the optimal bounds for perfectly secure MPC with graceful degradation of both security (allowing for hybrid security) and corruptions (allowing for mixed adversaries) for threshold as well as for general adversaries, given broadcast.[14] Furthermore, we show that the bounds are sufficient for MPC by providing parameters for the generalized protocols introduced in Sections 2 and 3, respectively. In the following section, we prove that the bounds are also necessary.

4.1 Threshold Adversaries

We consider a mixed adversary, which is characterized by a pair of thresholds (t_a, t_p): He may corrupt up to t_p parties passively, and up to t_a of these parties even actively. The level of security depends on the number $(|\mathcal{D}^*|, |\mathcal{E}^*|)$ 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. Depending on the actual number of corrupted parties, different security properties are achieved. This is modeled with four pairs of thresholds, one for each security requirement, specifying the upper bound on the number of corruptions that the adversary may perform, such that the security requirement is still guaranteed. More specifically, we consider the four pairs of thresholds (t_a^c, t_p^c), (t_a^s, t_p^s), (t_a^r, t_p^r), (t_a^f, t_p^f) and we assume that $(t_a^r, t_p^r) \leq (t_a^c, t_p^c)$ and $(t_a^f, t_p^f) \leq (t_a^s, t_p^s) \leq (t_a^c, t_p^c)$,[15] as secrecy and robustness are not well defined without correctness, and as fairness cannot be achieved without secrecy. Then, correctness with agreement on abort 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

[13] In particular $\forall S_i, S_j \in \mathcal{S} : S_i \cap S_j \neq \emptyset$, since otherwise no party can compute $a_i b_j$.
[14] The model without broadcast channels is treated in Appendix B.
[15] 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 $t_p^s \leq t_p^c$.

corresponding security properties are guaranteed. In particular, full security is guaranteed if the conditions for all four security properties are fulfilled.

Theorem 1. *In the secure channels model with broadcast and threshold adversaries, perfectly secure MPC among n 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.

Proof (Proof, Sufficiency). If $t_p^s = 0$, there is no secrecy requirement, and we can directly use the trivial non-secret protocol described in Appendix A. Otherwise, we employ the parametrized version $\pi^{d,e}$ of the protocol of [BGW88] described in Section 2 with $d := t_p^s$ and $e := \max(t_a^r, t_a^f)$.

The precondition of Lemma 7 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, note that $t_a^f \leq e$. Hence, for $(|\mathcal{D}^*|, |\mathcal{E}^*|) \leq (t_a^f, t_p^f)$ the protocol is robust, and the adversary cannot abort. □

The necessity of the bound in Theorem 1 is proven in Section 5. 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 $(t_a^f, t_p^f) = (t_a^s, t_p^s)$.

4.2 General Adversaries

The above characterization for threshold adversaries can be extended to general adversaries by providing one adversary structure consisting of tuples $(\mathcal{D}, \mathcal{E})$ of subsets of $\mathcal{P}$ for each security requirement, denoted by $\mathcal{Z}^c$, $\mathcal{Z}^s$, $\mathcal{Z}^r$, and $\mathcal{Z}^f$, respectively. Again, we have the assumption that $\mathcal{Z}^r \subseteq \mathcal{Z}^c$ and $\mathcal{Z}^f \subseteq \mathcal{Z}^s \subseteq \mathcal{Z}^c$, as secrecy and robustness are not well defined without correctness, and as fairness cannot be achieved without secrecy. Then, correctness with agreement on abort 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.

Theorem 2. *In the secure channels model with broadcast and general adversaries, perfectly secure MPC among n 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*

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

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

Proof (Proof, Sufficiency). If $\mathcal{Z}^s = \{(\emptyset, \emptyset)\}$, there is no secrecy requirement, and we can directly use the trivial non-secret protocol described in Appendix A. Otherwise, we employ the parametrized version $\pi^{\mathcal{S},\mathcal{C}}$ of the protocol of [Mau02] described in Section 3. We set $\mathcal{S} := \{\overline{\mathcal{E}^s} \mid (\cdot, \mathcal{E}^s) \in \mathcal{Z}^s\}$ and $\mathcal{C} = \{\mathcal{D} \mid (\mathcal{D}, \cdot) \in \mathcal{Z}^r \cup \mathcal{Z}^f\}$.

The precondition of Lemma 12 is that $\forall S \in \mathcal{S}, C_1, C_2 \in \mathcal{C} : S \not\subseteq C_1 \cup C_2$, which is equivalent to $\forall S \in \mathcal{S}, C_1, C_2 \in \mathcal{C} : \overline{S} \cup C_1 \cup C_2 \neq \mathcal{P}$. Note that $\mathcal{Z}^r \cup \mathcal{Z}^f \subseteq \mathcal{Z}^c$. Hence, for our choice of parameters $\mathcal{S}$ and $\mathcal{C}$, we have that for all S, C_1, and C_2, there are sets $\mathcal{E}^s$ and $\mathcal{D}^c$, such that $\overline{S} \cup C_1 \cup C_2 \subseteq \mathcal{E}^s \cup \mathcal{D}^c \cup C_2$. By construction of $\mathcal{C}$, $(C_2, \cdot)$ is either from $\mathcal{Z}^r$ or from $\mathcal{Z}^f$. Therefore, the inequality follows either from the left-hand side (if $(C_2, \cdot) \in \mathcal{Z}^r$) or from the right-hand side (if $(C_2, \cdot) \in \mathcal{Z}^f \subseteq \mathcal{Z}^s$) of the condition in the theorem. Hence, we can apply the lemma to derive correctness, secrecy and robustness: Given the bound in theorem, the choice of the structures $\mathcal{S}$ and $\mathcal{C}$, 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.

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

The necessity of the bound in Theorem 2 is proven in Section 5. Note that the bound holds for any $\mathcal{Z}^f \subseteq \mathcal{Z}^s$, and we can always have $\mathcal{Z}^f = \mathcal{Z}^s$.

5 Proofs of Necessity

In this section, we prove that the bounds in Theorem 1 and 2 are necessary, i.e., if violated, some (reactive) functionalities cannot be securely computed. Trivially, the impossibility for threshold adversaries follows from the impossibility for general adversaries. The bound for general adversaries (Theorem 2) is violated if

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

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 $\mathcal{D}^r = \emptyset$. Note that, 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 5.1)
   (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 5.2)
   (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 $|\mathcal{P}| \geq 2$, 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 5.3)
   (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 5.4)

5.1 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 3 (State). *A* state *for n parties $1, \ldots, n$ is a tuple $(s_1, \ldots, s_n)$ that defines a bit s, where party 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 13. *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 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 13, 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 13, which is a contradiction.

5.2 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 14. *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 chose between the state (a, b) and (a, b'), violating correctness. □

Given Lemma 14, we can prove the desired bound by reducing the n-party setting to the 2-party setting specified there, along the lines of the previous section.

5.3 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 15. *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 15, we can prove the desired bound by reducing the n-party setting to the 3-party setting specified there along the lines of the previous sections.

5.4 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 sections.

6 Conclusions and Open Problems

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), for both threshold and general adversaries. The protocols are strict generalizations (and combinations) of hybrid-secure MPC and mixed adversaries. We derived tight bounds for the existence of perfectly secure MPC protocols for the given settings, and provided protocols that 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 1 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 2). As a consequence, we provided the most comprehensive treatment, integrating the previous results on perfect security.

We leave as an open problem to combine additional dimensions of graceful degradation (like, e.g., efficiency) with graceful degradation of security and corruption types (e.g. fail-corruption), as well as to consider other security models (e.g. computational security). Furthermore, in this work, we focus on MPC including reactive functionalities. The bounds for secure function evaluation (SFE) might be slightly weaker.

References

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

[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-Verlag, 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.

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

[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-Verlag, 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-Verlag, 1989.

[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.

[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-Verlag, 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-Verlag, 1998.

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

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

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

[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*. Springer-Verlag, 2011.

[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-Verlag, 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-Verlag, 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.

[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-Verlag, 2002.

[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.

A A Trivial Non-Secret Protocol

If there is no secrecy requirement (i.e. $t_p^s = 0$, respectively $\mathcal{Z}^s = \{(\emptyset, \emptyset)\}$), each party can simply broadcast its inputs, and all parties locally compute the output.

Non-Secret Protocol

Input: Party i provides input s by broadcasting s to all parties.
Addition and Multiplication: Each party locally performs all computations on the broadcasted values.
Random values: Party 1 broadcasts a random value r.
Output: Each party simply outputs the value computed before.

Fig. 9. A protocol without secrecy.

Trivially, this protocol achieves correctness and robustness for any number of corrupted parties. As mentioned in Section 1.3, 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.

B The Model without Broadcast Channel

We now turn to a model where no broadcast channels are given, but only a complete network of pairwise secure channels. In this model, we obtain essentially the same bound as in the model with broadcast channels, with an additional limitation on the number of actively corrupted parties to at most one third of all parties.

Corollary 1. *In the secure channels model without broadcast and with threshold adversaries, perfectly secure MPC among n 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 2 on the perfectly secure broadcast protocol of [BGP89,CW89] tolerating $|\mathcal{D}^*| < \frac{n}{3}$ actively corrupted parties.

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 in presence of $|\mathcal{D}^*| \geq \frac{n}{3}$ actively corrupted parties [Lam83]. Thus, perfectly secure MPC without broadcast requires $3t_a^c < n$, in addition to the bound of Theorem 1.

An analogous statement holds for the general adversary case.

Corollary 2. *In the secure channels model without broadcast and with general adversaries, perfectly secure MPC among n 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 3 on the perfectly secure broadcast protocol of [FM98]. The necessity of the bound follows along the lines of the threshold case.