Multi-Threshold Asychronous Reliable Broadcast and Consensus

Multi-Threshold Asynchronous Reliable Broadcast and Consensus

Martin Hirt
Department of Computer Science, ETH Zürich, Switzerland
hirt@inf.ethz.ch

Ard Kastrati
Department of Information Technology and Electrical Engineering, ETH Zürich, Switzerland
akastrati@ethz.ch

Chen-Da Liu-Zhang
Department of Computer Science, ETH Zürich, Switzerland
lichen@inf.ethz.ch

Abstract

Classical protocols for reliable broadcast and consensus provide security guarantees as long as the number of corrupted parties f is bounded by a single given threshold t. If $f > t$, these protocols are completely deemed insecure. We consider the relaxed notion of *multi-threshold* reliable broadcast and consensus where validity, consistency and termination are guaranteed as long as $f \leq t_v$, $f \leq t_c$ and $f \leq t_t$ respectively. For consensus, we consider both variants of $(1-\epsilon)$-consensus and *almost-surely terminating* consensus, where termination is guaranteed with probability $(1-\epsilon)$ and 1, respectively. We give a very complete characterization for these primitives in the asynchronous setting and with no signatures:

- Multi-threshold reliable broadcast is possible if and only if $\max\{t_c, t_v\} + 2t_t < n$.
- Multi-threshold almost-surely consensus is possible if $\max\{t_c, t_v\} + 2t_t < n$, $2t_v + t_t < n$ and $t_t < n/3$. Assuming a global coin, it is possible if and only if $\max\{t_c, t_v\} + 2t_t < n$ and $2t_v + t_t < n$.
- Multi-threshold $(1-\epsilon)$-consensus is possible if and only if $\max\{t_c, t_v\} + 2t_t < n$ and $2t_v + t_t < n$.

2012 ACM Subject Classification Theory of computation → Cryptographic protocols; Theory of computation → Distributed algorithms; Security and privacy → Cryptography

Keywords and phrases broadcast, byzantine agreement, multi-threshold

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2020.6

Related Version A full version of the paper is available at `https://eprint.iacr.org/2020/958`.

This paper is eligible for best student paper award.

1 Introduction

Consensus and reliable broadcast are fundamental building blocks in fault-tolerant distributed computing. Consensus allows a set of parties, each holding an input, to agree on a common value v', where, if all honest parties hold the same input v, $v' = v$. Reliable broadcast allows a designated party, called the sender, to consistently distribute a value v among a set of recipients such that all honest recipients output v in case the sender is honest. If the sender is dishonest, either all honest recipients output the same value or none of them terminates. Both primitives are used typically in the design of more complex applications, including multi-party computation, verifiable secret-sharing or voting, just to name a few.

The first consensus protocol was introduced in the seminal work of Lamport et al. [21] for the model where parties have access to a complete network of point-to-point authenticated channels, and where at most $t < n/3$ parties are corrupted. Reliable broadcast was first

introduced by Bracha [6] as a useful primitive to construct building blocks in asynchronous environments. Since then, both primitives has been extensively studied in many different settings [7, 8, 6, 2, 25].

Most known fault-tolerant distributed protocols provide security guarantees in an *all-or-nothing* fashion: if up to t parties are corrupted, all security guarantees remain. However, if more than t parties are corrupted, the protocols do not provide any security guarantees. Multi-threshold protocols (also known as hybrid security) provide different security guarantees depending on the amount of corruption, thereby allowing a graceful degradation of security.

In this work, we consider consensus and reliable broadcast protocols with separate thresholds t_v, t_c and t_t for *validity*, *consistency* and *termination*, respectively. For consensus, we consider both variants of $(1-\epsilon)$-consensus and *almost-surely terminating* consensus, where termination is guaranteed with probability $(1-\epsilon)$ and 1, respectively.

Such multi-threshold primitives are not only of theoretical interest, but are also motivated by its use as core primitives in the design of more involved applications. In particular, they are used as a central building block in the recent line of works [26, 22], that leverage synchronous multi-party computation and consensus protocols to achieve *responsiveness*, where parties obtain output as fast as the network allows, given that the amount of corruption is low enough.

Our protocols work without the use of signatures and in the purely asynchronous model without the need to make any timing assumptions. Our contributions give a very complete picture of feasibility and impossibility results, which can be summarized as follows:

- Multi-threshold reliable broadcast is possible if and only if $\max\{t_c, t_v\} + 2t_t < n$.
- Multi-threshold almost-surely consensus is possible if $\max\{t_c, t_v\} + 2t_t < n$, $2t_v + t_t < n$ and $t_t < n/3$. The first two conditions are shown to be necessary as well. The question whether $t_t < n/3$ is necessary is left as an open problem. However, we give a protocol assuming a global coin that does not require this condition.
- Multi-threshold $(1-\epsilon)$-consensus is possible if and only if $\max\{t_c, t_v\} + 2t_t < n$ and $2t_v + t_t < n$.

The impossibility proofs are simple and follow the lines of [9].

1.1 Related Work

There is a large literature devoted to achieving different types of hybrid security guarantees under different settings for agreement primitives and multi-party computation (MPC). We are only able to list an incomplete summary of related work.

The work in [11] provides constructions in the synchronous model for Byzantine broadcast with extended validity or consistency, where Byzantine broadcast is achieved up to a threshold t, and validity / consistency is achieved up to an extended threshold $T \geq t$, and then apply such constructions to achieve multi-party computation with full security up to t corruptions, and *unanimous abort* up to $T \geq t$. The above constructions exists if and only if $t = 0$ or $t + 2T < n$. The works in [19, 20] focus on the question of achieving multi-party computation with full security under an honest majority, and *security with abort* under a dishonest majority. The line of works in [13, 24] provide constructions that achieve trade-offs that include information-theoretic security up to a certain threshold, and computational security up to a larger threshold, with different types of guarantees. A different line of works provide security against different types of corruption (also known as mixed adversaries). The works [12, 18] consider multi-party computation protocols where security holds even when up to t_p, t_f, t_a parties can be passively, fail-stop, actively corrupted, respectively. Finally, there are works that combine mixed adversaries with hybrid security [16, 17, 15].

A recent line of works [23, 26, 22] achieve trade-offs between *responsiveness*, where parties obtain output as fast as the network allows, and other security guarantees, for consensus, SMR and MPC, assuming a synchronous network. The work [14] considers to networks that tolerate some level of disconnection between the parties, as long as there is a connected component with an honest majority of the parties. Finally, the works [3, 4, 5] provide protocols that achieve security guarantees under a synchronous network up to t_s corruptions, and under an asynchronous network up to t_a corruptions.

1.2 Comparison to Prior Work

As mentioned above, some works use as building blocks multi-threshold asynchronous consensus and reliable broadcast primitives. In particular, the works in [23, 14, 22] make use of an asynchronous multi-threshold consensus protocol with increased validity and consistency. Their constructions differ from ours in two aspects: 1) they operate in a setting where parties have access to a public-key infrastructure and 2) their constructions inherently require that the termination threshold is below $n/3$.

The constructions for consensus and reliable broadcast in [3, 4] considers different thresholds. In [3], the authors design a consensus protocol with increased *validity with termination* (where validity also ensures termination in case of pre-agreement) assuming a global common coin, based on the protocol in [25]. Similarly, in [4], the authors provide a construction for reliable broadcast with two thresholds allowing for validity with termination in the honest sender case, and consistency with *reliable* termination (where either all honest parties terminate or none), in the dishonest sender case. We provide constructions without assuming a global coin, which in addition allow to have the termination threshold above validity and consistency.

2 Model

We consider a setting in which parties have access to a complete network of authenticated channels. The adversary has full control over the network and can schedule the messages in an arbitrary manner. However, each message must be eventually delivered. Moreover, we consider the setting where parties do not have any setup available.

We consider an *adaptive* adversary who can gradually corrupt parties and take full control over them. Note, however, that our impossibility proofs hold even against a *static* adversary that is assumed to choose the corrupted parties at the beginning of the protocol execution. We require our protocols to be *unconditionally secure*, meaning that security holds even against a computationally unbounded adversary. On the other hand, our impossibility results hold even against a computationally bounded adversary.

In the protocols we say that a party terminates when it stops participating in the protocol. Note that we distinguish between outputting and terminating, in the sense that a party might output a value but still continue participating.

3 Multi-Threshold Reliable Broadcast

Reliable broadcast is a fundamental primitive in distributed computing which allows a sender to consistently distribute a message towards a set of recipients. We consider a setting with $n+1$ parties, one sender S and n recipients $\mathcal{R} = \{R_1, ..., R_n\}$. Let us denote the number of corrupted recipients (not including the sender) at the end of the protocol execution by f.

▶ **Definition 1 (Reliable Broadcast).** *Let $\mathcal{M}$ be a finite message space and f be the number of corrupted recipients at the end of the execution. A protocol π where initially the sender S has an input $m \in \mathcal{M}$ and every recipient R_i upon termination outputs $m_i \in \mathcal{M}$, is a reliable broadcast protocol, with respect to thresholds t_c, t_v, and t_t, if it satisfies the following:*

- **Consistency**. *If $f \leq t_c$, then every honest recipient that terminates outputs the same message. That is, $\exists m' \in \mathcal{M} : \forall$ honest R_i that terminate $m_i = m'$.*
- **Validity**. *If $f \leq t_v$ and the sender is honest, then every honest recipient R_i that terminates outputs the sender's message. That is, $\forall$ honest R_i that terminate $m_i = m$.*
- **Termination**.
  1. *An honest sender always terminates.*
  2. *If $f \leq t_t$ and an honest recipient terminates, then every honest recipient eventually terminates.*
  3. *If $f \leq t_t$ and the sender is honest, then eventually every honest recipient terminates.*

3.1 Protocol

We present a reliable broadcast protocol with respect to thresholds t_c, t_v and t_t, as long as $\max\{t_v, t_c\} + 2t_t < n$. The protocol is a generalization of Bracha's broadcast protocol [6].

Protocol $\Pi_{\texttt{rbc}}^{t_c,t_v,t_t}$

The sender S holds input $m \in \mathcal{M}$. Upon termination every recipient $R_i \in \mathcal{R}$ outputs a message.

- **Code for the sender S**
  1. Send the message (MSG, m) to all recipients in $\mathcal{R}$ and terminate.
- **Code for recipient $R_i \in \mathcal{R}$**
  1. Upon receiving *first* (MSG, m) from the sender, send (ECHO, m) to all recipients.
  2. Upon receiving (ECHO, m') from $n - t_t$ parties that agree on the value $m' \in \mathcal{M}$, send (READY, m') to all recipients.
  3. Upon receiving (READY, m') from $\max\{t_v, t_c\} + 1$ parties that agree on the value $m' \in \mathcal{M}$, send (READY, m') to all recipients.
  4. Upon receiving (READY, m') or (TERMINATE) from $n - t_t$ parties from which at least $\max\{t_v, t_c\} + 1$ are READY messages and (they all) agree on the value $m' \in \mathcal{M}$, send (TERMINATE), output m' and terminate.

▶ **Lemma 2.** *If $f \leq \max\{t_v, t_c\}$, $\forall m' \in \mathcal{M}$ the first honest recipient that sends* (READY, m') *received at least $n - t_t$* (ECHO, m') *messages.*

Proof. For any $m' \in \mathcal{M}$, (READY, m') messages are sent by honest recipients either in line 2 or 3 of the recipient's code. However, $\forall m' \in \mathcal{M}$ the first (READY, m') message can only be sent in line 2. This is due to the fact that if $f \leq \max\{t_v, t_c\}$, line 3 implies that there must be some other honest recipient that previously sent a (READY, m') message too. ◀

▶ **Lemma 3.** *If $f \leq \max\{t_v, t_c\}$, the messages* (READY, m') *sent by honest recipients are consistent. That is, there $\exists m'' \in \mathcal{M}$ such that for every honest recipient R_i that sends* (READY, m')*, $m' = m''$.*

Proof. Suppose not; let R_i and R_j be the *first* honest recipients that send (READY, m') and (READY, m'') with $m' \neq m''$. Due to Lemma 2, R_i received at least $n - t_t$ (ECHO, m') messages whereas R_j received at least $n - t_t$ (ECHO, m'') messages. It follows, at least $2(n - t_t) - n = n - 2t_t > \max\{t_v, t_c\}$ players dishonestly sent inconsistent ECHO messages to R_i and R_j. However, each honest recipient sends an ECHO message at most once and there are at most $f \leq \max\{t_v, t_c\}$ dishonest recipients. A contradiction. ◀

▶ **Lemma 4.** *If $f \leq \max\{t_v, t_c\}$ and an honest sender broadcasts m, for every honest recipient that sends* (READY, m')*, $m' = m$.*

Proof. Suppose not; Let R_i be the first honest recipient that sends (READY, m') with $m' \neq m$. Due to Lemma 2, R_i received at least $n - t_t$ (ECHO, m') messages. However, in case the sender is honest and broadcasts m every honest recipient will ECHO only the sender's value (m). Hence there can be at most $f \leq \max\{t_v, t_c\} < n - t_t$ (ECHO, m') with $m \neq m'$. A contradiction. ◀

▶ **Lemma 5.** *If $f \leq t_t$ and an honest recipient terminates, then every honest recipient eventually terminates.*

Proof. Let R_i be the first honest recipient that terminates. Then, R_i received (READY, m') or (TERMINATE) from $n - t_t$ messages from which at least $\max\{t_v, t_c\} + 1$ are READY messages. Furthermore, all READY messages agree on m'. Under the assumption that no other honest recipient has terminated so far, we know that no (TERMINATE) messages were sent from the honest recipients. Hence, by taking into account that $n - t_t > \max\{t_v, t_c\} + t_t$ and $f \leq t_t$, it follows that at least $\max\{t_v, t_c\} + 1$ recipients have sent (READY, m') to all other parties. Every honest recipient will eventually either receive these (READY, m') messages and send a (READY, m') as well or terminate before receiving them and send a (TERMINATE) message instead. Since there are at least $n - t_t$ honest recipients, it follows that eventually every honest recipient R_l that didn't terminate yet will receive $n - t_t$ messages (READY, m') or (TERMINATE) from which at least $\max\{t_v, t_c\} + 1$ are READY messages and they all agree on m'. Thus, every honest R_l eventually terminates as well. ◀

▶ **Lemma 6.** *If $f \leq t_t$ and the sender is honest, at least one honest recipient eventually terminates.*

Proof. Since every honest recipient echoes the sender's value, there will be at least $n - t_t$ (ECHO, m) messages. Similarly, since there are $n - t_t$ (ECHO, m) messages in the network, every honest recipient will eventually send a (READY, m). Finally, since there are $n - t_t$ (READY, m) messages in the network, at least one honest recipient will terminate. ◀

▶ **Theorem 7.** *Let $0 \leq t_c, t_v, t_t < n$. $\Pi_{\texttt{rbc}}^{t_c,t_v,t_t}$ is a multi-threshold broadcast according to Definition 1 if $\max\{t_v, t_c\} + 2t_t < n$.*

Proof.

- *Consistency & Validity.* Assume $f \leq \max\{t_v, t_c\}$. Every honest recipient that outputs a message, has received at least $\max\{t_v, t_c\} + 1$ (READY, m). Since $f \leq \max\{t_v, t_c\}$, it follows at least one is sent from an honest party. From Lemma 3 we know that READY messages are consistent, hence we achieve consistency. From Lemma 4 we know that the READY messages from honest parties contain only the sender's value, hence we achieve validity.
- *Termination.* We prove the three termination properties from the Definition 1.
  1. For the first requirement, it is trivial to see that an honest sender always terminates.
  2. The second requirement is proven in Lemma 5.
  3. For the third requirement, from Lemma 6 we know that if the sender is honest, at least one honest recipient terminates. We can apply Lemma 5 again, and see that every honest recipient terminates. ◀

3.2 Impossibility Proofs

In this section, we show that the protocol $\Pi_{\texttt{rbc}}^{t_c,t_v,t_t}$ presented above is optimal. That is, there is no reliable broadcast protocol with $\max\{t_v, t_c\} + 2t_t \geq n$, where $t_c, t_v, t_t \geq 0$. We prove each bound separately.

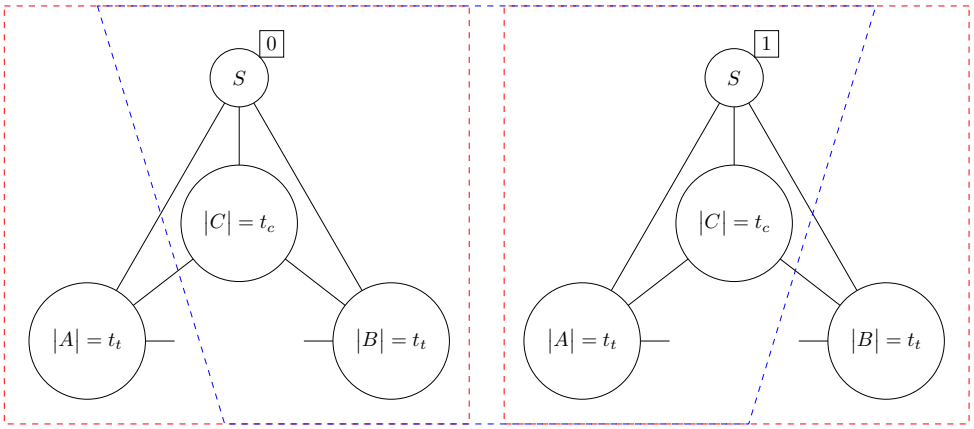

Figure 1 The same configuration can be viewed as: a) (in red) two independent runs of the protocol on the left and the right side, where messages between A and B are delayed: One where the sender has input 0 and one where the sender has input 1; b) (in blue) the sender and the set C is corrupted and behaves differently towards A and B.

▶ **Theorem 8.** *There exists no multi-threshold broadcast protocol with $t_c + 2t_t \geq n$.*

Proof. Suppose not; let π be a multi-threshold broadcast protocol with $t_c + 2t_t = n$. We partition the set of all recipients into three sets A, B and C with size $|A| = |B| = t_t$ and $|C| = t_c$. We build the network as in Figure 1.

Figure 1(in red) on the left side, we can see an independent run where all parties are honest and messages between A and B are delayed by the scheduler. Since A and B are of size t_t, all parties terminate with an output. Moreover, since all parties are honest, the output is 0. In particular, parties in A output 0. Similarly, B outputs 1 on the right side.

Now consider an attacker that corrupts the sender and C, and emulates the protocol as in the scenario in Figure 1(in blue). Since this configuration is exactly the same as the red one, A outputs 0 and B outputs 1. This results in a contradiction to the consistency property of the multi-threshold broadcast. ◀

▶ **Theorem 9.** *For any $t_v > 0$, there is no multi-threshold broadcast protocol with $t_v + 2t_t \geq n$.*

Proof. Suppose not; let π be a multi-threshold broadcast protocol with $t_v + 2t_t = n$. We partition the set of all recipients into three sets A, B and C with $|A| = |B| = t_t$ and $|C| = t_v$. We build the a configuration as in Figure 2.

Consider Figure 2(in red), on the left side, where messages between S and B, or between A and B are delayed. Since B is of size t_t, all parties must output a value *without* waiting for the messages from B (as B could be corrupted). Moreover, since all parties are honest, A and C output 0. Furthermore, since C outputs 0, because of the second requirement of the termination of broadcast – *if one recipient terminates, then every recipient terminates* – B outputs 0 as well. Note that A and S have together size $t_t + 1$, but the second requirement of termination requires B to terminate even if the sender is corrupted. The same argument can be applied on the right side of Figure 2(in red).

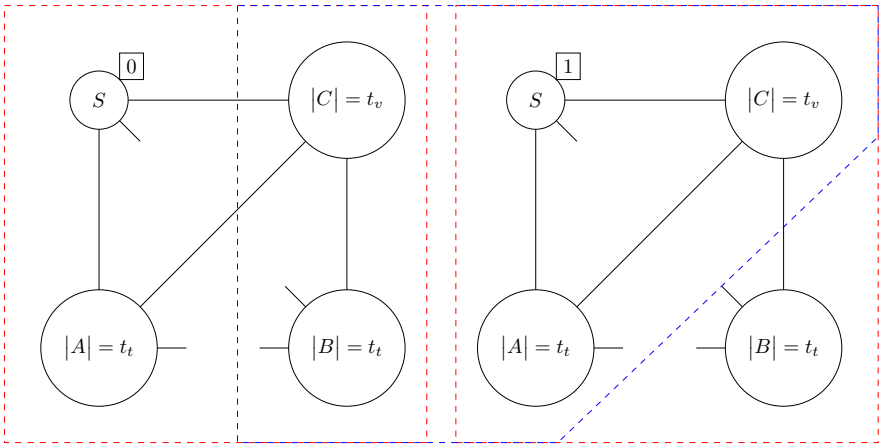

Figure 2 The same configuration can be viewed as: a) (in red) two independent runs of the protocol on the left and the right side, where messages between A and B are delayed, and also between S and B. One where the sender has input 0 and one where the sender has input 1; b) (in blue) the set C is corrupted and behaves differently towards S and A on the left side and differently towards B on the right side.

Now, consider an attacker that corrupts the parties in C and emulates the protocol as in the scenario in Figure 2(in blue). Because both scenarios are the same setup, A outputs 0 on the left side, whereas B outputs 1 on the right side. Thus, validity is violated. ◀

4 Almost-Surely Multi-Threshold Consensus

Stated in simple terms, consensus allows a set of parties to agree on a common value. More formally, the protocol starts with every party having an input and ends with every party having a consistent output. Moreover, if every honest party starts with the same input, they keep it. Due to the FLP impossibility proof [10], non-terminating executions are inevitable for every consensus protocol. Hence, we require the parties to terminate only with probability 1, termed in the literature as *almost-surely terminating* consensus.

▶ **Definition 10 (Consensus).** *Let $\mathcal{M}$ be a finite message space and f be the number of corrupted parties at the end of the execution. A protocol π where initially each party has an input $x_i \in \mathcal{M}$ and finally every party P_i upon termination has an output $y_i \in \mathcal{M}$, is a consensus protocol, with respect to thresholds t_c, t_v, t_t, if it satisfies the following:*

- **Consistency**. *If $f \leq t_c$, then the output of every honest party is the same value. That is, $\exists y \in \mathcal{M} : \forall$ honest P_i that output $y_i = y$.*
- **Validity**. *If $f \leq t_v$ and every honest party has the same input value $x \in \mathcal{M}$, then the output of every honest party P_i is x. That is, $\forall$ honest P_i that output $y_i = x$.*
- **Termination**. *If $f \leq t_t$, then with probability 1 eventually every honest party outputs and terminates.*

In the following, we present a multi-threshold consensus protocol with respect to thresholds t_c, t_v and t_t, where $\max\{t_c, t_v\} + 2t_t < n$, $2t_v + t_t < n$ and $t_t < n/3$. In the full version, we also show that the bounds $\max\{t_c, t_v\} + 2t_t < n$ and $2t_v + t_t < n$ are required. We leave the feasibility of almost-surely multi-threshold consensus with $t_t \geq n/3$ as an open question. However, in the full version we provide a construction that overcomes the $n/3$ bound for the case where parties have access to a global coin.

The protocol is an adaptation of Bracha's consensus [6] protocol. The main idea of Bracha's consensus is to use a reliable broadcast primitive and build a *correctness enforcement* scheme, where only messages that are intended by the protocol design are accepted. The only difference in our protocol is that we plug our multi-threshold broadcast protocol described in the previous section into the correctness enforcement scheme proposed by Bracha. We choose to use a multi-threshold reliable broadcast $\Pi_{\texttt{rbc}}^{t_s,t_s,t_t}$ that achieves validity and consistency up to $t_s = n - 2t_t - 1$ corruptions (which is achievable since $t_s + 2t_t < n$). Note that $t_t < n/3$ implies that $t_s \geq t_t$.

4.1 Multi-Threshold Correctness Enforcement

For completeness and readability of our protocols, we include a summary of the correctness enforcement mechanism. Further details can be found in [6]. The following constructions and proof techniques are very similar to [6] with the only difference that we plug our multi-threshold broadcast protocol $\Pi_{\texttt{rbc}}^{t_s,t_s,t_t}$. Furthermore, we assume $t_s \geq t_t$.

Round-Based Asynchronous Protocols. We consider protocols that are composed by *rounds*. In each round k, every party uses the multi-threshold broadcast protocol to send a value to all parties. Subsequently, every party waits to receive a set S of the values (of size at most $n - t_t$) and computes a new value according to some function $F^k(\cdot)$ for the next round.

Validation Sets. Each party P_i keeps for each round k a set of values $\mathcal{V}_i^k$, called a *validation set*, with the values that are broadcast in round k. Each value x_i broadcasted in round k by P_i is stored as (P_i, k, x_i). When a value is broadcast by some party at round $k+1$, every party checks locally whether there exists a subset of values in $\mathcal{V}_i^k$ that explains the broadcast value. That is, $\mathcal{V}_i^k$ is defined as follows:

- For $k = 1$, $(P_j, 1, x_j) \in \mathcal{V}_i^1$ if x_j is received by P_i from a multi-threshold broadcast protocol with sender P_j at round 1.
- For $k > 1$, $(P_j, k, x_j) \in \mathcal{V}_i^k$, if x_j is received by P_i from a multi-threshold broadcast protocol with sender P_j at round k, and there is a subset $S \subseteq \mathcal{V}_i^{k-1}$ such that $x_j = F^{k-1}(S)$.

We say a party P_i *validates* a message x_j in round k if $(P_j, k, x_j) \in \mathcal{V}_i^k$. The parties update their $\mathcal{V}$ sets whenever they validate a message. If a party P_i outputs a value during a broadcast protocol but the message is still not validated it is ignored in the protocol, although it is stored for future validation.

Protocol A round with correctness enforcement

Code for the party P_i with input x_i at round k

1. Multi-Threshold Broadcast(x_i) to all the parties.
2. Wait until a set S of messages have been validated.
3. Set $x_i = F^k(S)$.

We state a list of lemmas that are guaranteed from the correctness enforcement mechanism. The proofs will appear in the full version of the paper.

▶ **Lemma 11.** *If $f \leq t_s$, in every round k of the protocol the added values in the validation sets of all honest parties are consistent. That is:*

$$\forall \text{ honest } P_i, P_j : \forall P_l : (P_l, k, x_l) \in \mathcal{V}_i^k \wedge (P_l, k, \tilde{x}_l) \in \mathcal{V}_j^k \implies x_l = \tilde{x}_l$$

▶ **Lemma 12.** *If $f \leq t_s$ and an honest sender P_i broadcasts (P_i, k, x_i) in a round k of the protocol, the validation sets of all honest parties contain only the sender's value. That is:*

$$\forall \text{ honest } P_i, P_j : (P_i, k, x_i) \in \mathcal{V}_j^k \implies P_i \text{ broadcast } x_i \text{ in round } k.$$

▶ **Lemma 13.** *If $f \leq t_t$ every party will eventually go from round k to round $k+1$.*

4.2 Protocol

We describe the protocol, which is a generalization of Bracha's consensus [6]. The protocol executes in parallel the two sub-protocols 'Reaching agreement' and 'Termination'.

Protocol $\Pi_{\text{as-con}}^{t_c,t_v,t_t}$

Input. Every party P_i holds input x_i.
Variable. $y_i = \perp$.

Reaching agreement.

- **Code for the party P_i at phase k.**
  1. $\Pi_{\text{rbc}}^{t_s,t_s,t_t}(x_i)$. Wait until $n - t_t$ messages are validated.
     - x_i = majority of the validated elements.
  2. $\Pi_{\text{rbc}}^{t_s,t_s,t_t}(x_i)$. Wait until $n - t_t$ messages are validated.
     - If all of the validated messages have the same value x, $x_i = (\text{propose}, x)$
     - Otherwise, keep the same x_i.
  3. $\Pi_{\text{rbc}}^{t_s,t_s,t_t}(x_i)$. Wait until $n - t_t$ messages are validated.
     - **a.** If at least $n - t_t$ of the validated messages have the same value $(\text{propose}, x)$, then update $y_i = x$ and run the 'Reaching agreement code' for only one more phase.
     - **b.** Else if at least $t_t + 1$ of the validated messages have the same value $(\text{propose}, x)$, then $x_i = x$.
     - **c.** Otherwise, choose 0 or 1 with probability 1/2 for x_i (coin toss).
  4. Go to phase $k + 1$.

Termination.

- Upon updating y_i, send (READY, y_i) to all parties.
- Upon receiving (READY, m') messages from $\max\{t_c, t_v\} + 1$ parties that agree on the value m', send (READY, m') to all parties.
- Upon receiving (READY, m') from $n - t_t$ parties that agree on the value m', output m' and terminate.

▶ **Lemma 14.** *If $f \leq t_s$, it is impossible for an honest party to propose 0 and an honest party to propose 1 in the same phase k.*

Proof. The proof is by contradiction. Suppose parties P_i and P_j propose 0 and 1, respectively, in phase k. Thus in line 2 of phase k, P_i validated $n - t_t$ messages with value 0 and P_j validated $n - t_t$ messages with value 1. Since $n - 2t_t > 0$, it follows that P_i and P_j have inconsistent messages in their validation sets, which contradicts Lemma 11. ◀

We say that a phase k is x-fixed (for $x \in \{0, 1\}$), if honest parties that starts phase k validate only x as an input value broadcast by any party.

▶ **Lemma 15.** *If $f \leq t_s$ and an honest party P_i updates $y_i = x \in \{0, 1\}$ at some phase k, phase $k + 1$ is x-fixed.*

Proof. Suppose that some party P_i updates $y_i = x \in \{0, 1\}$ at phase k. P_i must have validated at least $n - t_t$ proposals for x in step 3 of phase k. Let P_j be any party that starts phase $k + 1$. In phase k, since P_i validated $n - t_t$ proposals for x and $t_t < n/3$, P_j must have

validated at least $t_t + 1$ for x. Moreover by Lemma 14, P_j does not validate any proposals for $x' \neq x$. So any P_j can only set its variable x_j to x in step 3. Hence, no honest party can validate $x' \neq x$ in the next phase as an input. Therefore, phase $k + 1$ is x-fixed. ◀

▶ **Lemma 16.** *If a phase k is x-fixed then every honest party P_i that reaches step 3 of the phase k updates $y_i = x$ at the end of the phase.*

Proof. Suppose a phase k is x-fixed. Then, all honest parties validate only x as an input value. Hence, every honest party can only propose x as their input. Consider a party P_i that reaches step 3 of phase k. Clearly, P_i can only validate proposals with x. Hence, from line 3 of the *reaching agreement* part of the protocol we can see that P_i updates $y_i = x$. ◀

▶ **Lemma 17** (Liveness)**.** *If $f \leq t_t$ and if no honest party updated y_i, the parties will eventually go from phase k to phase $k + 1$.*

Proof. Immediate from Lemma 13. ◀

▶ **Lemma 18.** *If $f \leq t_t$ every honest party with probability 1 will update y_i to the same value.*

Proof. First note that we assume $t_t \leq t_s$. By Lemma 17, as long as no party updates y_i, honest parties don't get stuck in any round. Every honest party that doesn't update y_i in phase k, sets its value x_i for the next phase either based on step 3(ii) or step 3(iii). Let P_i be the first honest party that completed round $3k + 3$. There are two cases:

- Party P_i has validated at least one $(\text{propose}, x)$. With probability $p \geq 1/2^{n-t_t}$ all honest parties that toss a coin choose x. By Lemma 14, the remaining honest parties that set their value deterministically, are forced to set their value to x.
- Party P_i has validated no $(\text{propose}, x)$. Since P_i validated $n - t_t$ messages and $t_t < n/3$, no other honest party P_j can validate more than t_t values of the form $(\text{propose}, x)$. Hence, every honest party tosses a coin. The probability that every honest party tosses the same value is again $p \geq 1/2^{n-t_t}$.

Hence, in either case after each phase the probability that honest parties have the same value is greater or equal then $1/2^{n-t_t}$. If at some phase k every honest party has the same value then it follows that in the next round there can be at most t_t parties will input $\bar{x}$ (the ones that maliciously change the outcome of the local coin). However, by Lemma 15 (note that we have $t_t \leq t_s$) and since $n - 2t_t > t_t$, it follows that the majority of each subset of size $n - t_t$ in the next round will result in x. Hence, next phase is x-fixed. By Lemma 16 it follows that every honest party will update y_i after that phase. Hence, after round k the probability of not updating y_i is $(1 - p)^k$. As k goes to infinity, the probability goes to 0. ◀

▶ **Theorem 19.** *Let $0 \leq t_c, t_v, t_t < n$. $\Pi^{t_c,t_v,t_t}_{\text{as}-\text{con}}$ is an almost-surely terminating multi-threshold consensus (see Definition 10) if $\max\{t_c, t_v\} + 2t_t < n$, $2t_v + t_t < n$ and $t_t < n/3$.*

Proof.

- *Validity.* Suppose all honest parties have the same input x. In the first round, since there are at most $f \leq t_v$, at most t_v elements with value $\bar{x}$ can be broadcast by corrupted parties. By Lemma 15 (note that $t_v \leq \max\{t_c, t_v\} = t_s$) and since $n - t_t - t_v > t_v$, it follows that honest parties can only validate x as the outcome of the first round. Hence, the first phase is x-fixed. Due to Lemma 16, it follows that every honest party updates $y_i = x$ at the end of the first phase. Furthermore, by applying Lemma 15 and 16 recursively, one can easily see that once a phase is x-fixed it always remains so. Hence, parties can never change the value. Furthermore, in the termination part of the protocol it is not hard to see that READY messages are unique (see Section 3 for a detailed proof) and contain only x. Hence, every honest party that outputs, outputs x.

- *Consistency.* Suppose some party P_i and P_j update different values $y_i = x$ and $y_j = x'$ in phase k and k' respectively. There are two cases:
  1. $k = k'$. Since a party can update a value in phase k only if that value was proposed in round k, it follows both x and x' were proposed in phase k. By Lemma 14 this is impossible.
  2. $k < k'$. Since P_i updates y_i in phase k then due to Lemma 15 phase $k+1$ is x-fixed. Similar to previous arguments, by applying Lemma 15 and 16 recursively, one can easily see that once a phase is x-fixed it always remains so. It follows that all parties can only update $y_i = x$ in the next phases.

  This proves that the update of y_i by all parties is unique. In return this implies that READY messages are unique as well. By similar arguments as in Section 3, it is not hard to see that the consistency is achieved also in the termination part of the protocol.
- *Almost-Surely Termination.* From Lemma 18 it follows that with probability 1 every honest party will update y_i to the same value (say x). Since, after updating $y_i = x$ parties take part only in one more round, with probability 1 every honest party will "get out" of the infinite loop. By simply setting the message space $\mathcal{M} = \{0, 1\}$ one can easily prove now termination similar to Section 3. It follows, like in multi-threshold broadcast protocol, every party will eventually send the same (READY, x) with $x \in \{0, 1\}$. Hence eventually there will be $n - t_t$ (READY, x) messages that agree on a value and thus at least an honest party with probability 1 terminates. Again, similar to broadcast one can see that if an honest party terminates, then every honest party eventually terminates. Thus with probability 1 eventually everyone terminates. ◀

5 $(1 - \epsilon)$ Multi-Threshold Consensus

The general idea in the previous section is to use randomness such that by chance the parties reach agreement. Once they do, agreement is preserved. However, in the regime where $t_t \geq n/3$, the following challenges arise. First, note that if $t_t \geq n/3$ and $\max\{t_c, t_v\} + 2t_t < n$, then $\max\{t_c, t_v\} < t_t$. As a consequence, there is a region where the multi-threshold broadcast protocol only guarantees termination, but does not guarantee the consistency and validity of the outputs. This is problematic, because the adversary can change the outputs of each broadcast instance such that no messages are validated in the correctness enforcement scheme. As a consequence, parties get stuck in a phase and never terminate. The second challenge is with respect to the coin. If $t_t \geq n/3$, even when all honest parties obtain the same value v as local coin, the adversary can schedule messages so that the majority decision among $n - t_t$ values is inconsistent among the parties. Finally, as pointed out in [1], the correctness proof for $n/3 \leq t_t < n/2$ is more subtle and requires reasoning about two consecutive phases. Moreover, they show that the global-variant of Ben-Or *doesn't work* for the case $n/3 \leq t_t < n/2$.

We overcome the first challenge by plugging in a *detectable broadcast* primitive into the correctness enforcement mechanism, which allows parties to eventually detect misbehavior in the case where they obtain different values. The second challenge is resolved by cycling through all sets S of $t_t + 1$ parties, where only parties in S sample a random coin. This way, if all parties in S are honest and chooses the same local coin, then everyone adopts the same value. Finally, the last challenge is resolved by adding one round for each phase, which allows to analyse the phases independently of each other.

With these techniques, we construct a $(1-\epsilon)$ multi-threshold consensus protocol if $\max\{t_c, t_v\} + 2t_t < n$ and $2t_v + t_t < n$, where termination holds with probability $(1-\epsilon)$ (instead of 1). The bounds are optimal (see the full version).In contrast to the almost-surely-terminating version, it is possible to achieve this notion with $t_t \geq n/3$. The number of rounds depends on the error ϵ. For negligible ϵ, the number of rounds is exponential in n.

▶ **Definition 20** ($(1-\epsilon)$-Consensus)**.** *The consistency and validity property of* $(1-\epsilon)$*-consensus are the same as in Definition 10. We only change the termination property.*

- **Termination**. *If $f \leq t_t$, then with probability $1-\epsilon$ eventually every honest party outputs and terminates.*

5.1 Detectable Correctness Enforcement

As mentioned, if $t_t > \max\{t_v, t_c\}$, in the region where the adversary corrupts $\max\{t_v, t_c\} < f \leq t_t$, the multi-threshold broadcast does not guarantee any consistency among messages, allowing the adversary to make parties reach a state where no messages are validated and thus all honest parties get stuck. Instead, we use a *detectable* multi-threshold broadcast, which guarantees consistent outputs when $f \leq \max\{t_c, t_v\}$ as in multi-threshold broadcast, but in addition allows parties to detect potential misbehavior if $f \leq t_t$. Note, however, that we don't require termination, i.e., parties may need to run forever. The protocol is based on the one in Section 3, so we defer its description and analysis to the full version. Plugging the detectable multi-threshold broadcast in the correctness enforcement results in detectable correctness enforcement, where the properties of correctness enforcement hold or parties detect Byzantine behavior.

▶ **Definition 21** (**Detectable Multi-Threshold Broadcast**)**.** *Let $\mathcal{M}$ be a finite message space and f be the number of corrupted parties. A protocol π, where initially the sender S has an input message $m \in \mathcal{M}$ and subsequently every recipient $R_i \in \mathcal{R}$ potentially outputs a message $m_i \in \mathcal{M}$ and/or a detection flag* DETECT*, is a detectable multi-threshold broadcast protocol with respect to thresholds t_c, t_v and t_t, if it satisfies the following:*

- **Consistency.** *If $f \leq t_c$, $\exists m' \in \mathcal{M} : \forall$ honest R_i that output the message m_i, the value of $m_i = m'$. Furthermore, no honest recipient R_i outputs the detection flag* DETECT.
- **Validity.** *If $f \leq t_v$ and the sender is honest, $\forall$ honest R_i that output the message m_i, the value of $m_i = m$. Furthermore, no honest recipient R_i outputs the detection flag* DETECT.
- **Totality-or-Detection**.
  1. *If $f \leq t_t$ and an honest recipient outputs the message $m' \in \mathcal{M}$ then eventually every honest recipient outputs the message m' or every honest recipient outputs the detection flag* DETECT.
  2. *If $f \leq t_t$ and the sender is honest, then eventually every honest recipient outputs the message m or every honest recipient outputs the detection flag* DETECT.

Notation. We say that "P_i detects Byzantine behaviour" to denote that P_i outputs the detection flag DETECT in an execution of detectable multi-threshold broadcast. Note that detectable multi-threshold broadcast guarantees that either all honest recipients eventually output DETECT, or none of them does.

5.2 Common Coin

In the protocol from Section 4, parties toss a coin until they reach by chance agreement. If $t_t \geq n/3$, the adversary can maliciously change the local coins for some of the parties and break termination. We show a protocol that allows parties to output the same value, even if

the adversary changes the outcome of the local coin of some of the parties. If $t_t < n/3$, this is easily done by allowing each party to broadcast a random value and choosing the majority among the $n - t_t$ values that are received. For $n/3 \leq t_t < n/2$, one cannot take the majority value. The idea is to let only a subset R of $t_t + 1$ parties toss a local coin:

Protocol Coin(R), $f \leq t_t < n/2$, $|R| = t_t + 1$

1. If $P_i \in R$: choose 0 or 1 with probability 1/2 and detectable-broadcast the outcome to everyone.
2. Every party outputs the value of the first broadcast that output.

If all the parties in R are honest and toss the same coin, then every party outputs the same value. During the consensus protocol, we *cycle* through all subsets of size $t_t + 1$. If $t_t < n/2$, one of them contains only honest parties, and in that phase, if all parties toss the same coin (we denote it a *lucky* phase), all honest parties obtain the same value.

5.3 Protocol

The protocol is very similar to the one in Section 4, but with four changes: 1) it is executed a fixed number of phases, 2) the broadcast protocols are replaced by detectable broadcast protocols (allowing parties for detectable correctness enforcement), 3) the coin is replaced by the one in Section 5.2 and 4) a termination protocol which works even if $t_t > \max\{t_c, t_v\}$. In addition, the protocol has a special initial majority round that allows for a simpler analysis of validity and one *lock-round* for each phase that allows the deterministic value of a phase to be fixed before the coins are revealed.

The intuition behind fixing the number of phases is that if the protocol runs indefinitely, it may happen that some messages are *never* scheduled: even though the detectable broadcast *eventually* detects misbehavior, such messages are never scheduled because there are always other messages that the adversary can schedule[1]. However, this cannot happen if the number of phases is fixed. Setting a "large enough" number of phases suffices for parties to reach agreement with probability $(1-\epsilon)$, unless the adversary misbehaved in a detectable broadcast protocol, in which case parties detect it and eventually reach agreement.

We call a batch B an iteration over all subsets of size $t_t + 1$, i.e. $B = \binom{n}{t_t+1}$. We set an upper bound $K + 1$ on the number of batches (hence we have $(K + 1)B$ phases in total), so that the probability that parties are not in agreement after $(K + 1)B$ phases is at most ϵ.

Protocol $\Pi^{t_c,t_v,t_t}_{(1-\epsilon)-\texttt{con}}$

Input. Every party P_i holds input $x_i \in \{0, 1\}$.
Variable. $y_i = \perp$.
Initial majority round *// This initial round is necessary to ensure validity.*
- Detectable-Broadcast(x_i) to every party. Wait until $n - t_t$ messages have been validated.
  - Set x_i = majority of the $n - t_t$ validated messages.

Reaching agreement. Repeat at most $K + 1$ times: *// $K + 1$ batches.*
- For every subset R of size $|R| = t_t + 1$ do: *// we call this loop one batch.*
  1. Detectable-Broadcast(x_i) to every party. Wait until $n - t_t$ messages have been validated.

[1] This is the main challenge that one needs to overcome to design an almost-surely terminating consensus for $t_t \geq n/3$.

- If all validated messages have the same value x, update $x_i = (\text{lock}, x)$, else update $x_i = (\text{lock}, ?)$.

2. Detectable-Broadcast(x_i) to every party. Wait until $n - t_t$ messages have been validated.
   - If all messages are (lock, x) with the same $x \in \{0, 1\}$, update $x_i = (\text{propose}, x)$, else update $x_i = (\text{propose}, ?)$.
3. Set $c_i = \text{Coin}(R)$.
4. Detectable-Broadcast(x_i) to every party. Wait until $n - t_t$ messages have been validated.
   - If all messages are $(\text{propose}, x)$ with $x \in \{0, 1\}$, update $y_i = x$.
   - Else if at least one of the messages is $(\text{propose}, x)$ with $x \in \{0, 1\}$, then $x_i = x$.
   - Otherwise, set $x_i = c_i$.

Termination.

- Upon updating y_i, send (READY, y_i) to all parties.
- Upon detecting a Byzantine behaviour, send (READY, $\perp$) to all parties.
- Upon receiving (READY, d') messages from $\max\{t_c, t_v\} + 1$ parties that agree on the value $m' \in \{0, 1, \perp\}$ during the consensus protocol, send (READY, m') to all parties.
- Upon receiving (READY, m') or (TERMINATE) from $n - t_t$ parties from which at least $\max\{t_v, t_c\} + 1$ are READY messages and (they all) agree on the value $m' \in \{0, 1, \perp\}$, send (TERMINATE) to all recipients, output m' and terminate.

A formal analysis of the protocol can be found in the full version of the paper. Intuitively, if $f \leq \max\{t_c, t_v\}$, correctness enforcement ensures the same properties as in the protocol $\Pi^{t_c,t_v,t_t}_{\texttt{as-con}}$, making the proofs of validity and consistency similar to those. However, the termination property involves a bit more careful analysis. The idea is that either each honest party P_i updates to the same value $y_i = y$, or Byzantine behavior is detected. This is because with high probability, there is a phase where honest parties reach agreement by chance (they obtain the same value from the coin, and the coin coincides with the deterministic value of that phase), unless the adversary tampered the outputs from a detectable broadcast protocol in which case it will eventually be detected. As a result, one can argue that there is an honest party that terminates (every honest party eventually sends the same READY message), which in turn implies that eventually everyone terminates by a similar argument as for the broadcast protocol in Section 3.

References

1 Marcos K. Aguilera and Sam Toueg. The correctness proof of Ben-Or's randomized consensus algorithm. *Distributed Computing*, 25(5):371–381, 2012. doi:10.1007/s00446-012-0162-z.

2 Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In *Proceedings of the Second Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 27–30. ACM, 1983. doi:10.1145/800221.806707.

3 Erica Blum, Jonathan Katz, and Julian Loss. Synchronous consensus with optimal asynchronous fallback guarantees. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 131–150. Springer, Heidelberg, December 2019. doi:10.1007/978-3-030-36030-6_6.

4 Erica Blum, Jonathan Katz, and Julian Loss. Network-agnostic state machine replication. Cryptology ePrint Archive, Report 2020/142, 2020. URL: https://eprint.iacr.org/2020/142.

5 Erica Blum, Chen-Da Liu-Zhang, and Julian Loss. Always have a backup plan: Fully secure synchronous mpc with asynchronous fallback. In Daniele Micciancio and Thomas Ristenpart,

editors, *Advances in Cryptology - CRYPTO 2020*, pages 707–731, Cham, 2020. Springer International Publishing.

6 Gabriel Bracha. Asynchronous Byzantine agreement protocols. *Information and Computation*, 75(2):130–143, 1987. doi:10.1016/0890-5401(87)90054-X.

7 Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

8 Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997.

9 Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. Easy impossibility proofs for distributed consensus problems. In Michael A. Malcolm and H. Raymond Strong, editors, *4th ACM PODC*, pages 59–70. ACM, August 1985. doi:10.1145/323596.323602.

10 Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985. doi:10.1145/3149.214121.

11 Matthias Fitzi, Martin Hirt, Thomas Holenstein, and Jürg Wullschleger. Two-threshold broadcast and detectable multi-party computation. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 51–67. Springer, Heidelberg, May 2003. doi:10.1007/3-540-39200-9_4.

12 Matthias Fitzi, Martin Hirt, and Ueli M. Maurer. Trading correctness for privacy in unconditional multi-party computation (extended abstract). In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 121–136. Springer, Heidelberg, August 1998. doi:10.1007/BFb0055724.

13 Matthias Fitzi, Thomas Holenstein, and Jürg Wullschleger. Multi-party computation with hybrid security. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 419–438. Springer, Heidelberg, May 2004. doi:10.1007/978-3-540-24676-3_25.

14 Yue Guo, Rafael Pass, and Elaine Shi. Synchronous, with a chance of partition tolerance. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 499–529. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26948-7_18.

15 Martin Hirt, Christoph Lucas, and Ueli Maurer. A dynamic tradeoff between active and passive corruptions in secure multi-party computation. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 203–219. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1_12.

16 Martin Hirt, Christoph Lucas, Ueli Maurer, and Dominik Raub. Graceful degradation in multi-party computation (extended abstract). In Serge Fehr, editor, *ICITS 11*, volume 6673 of *LNCS*, pages 163–180. Springer, Heidelberg, May 2011. doi:10.1007/978-3-642-20728-0_15.

17 Martin Hirt, Christoph Lucas, Ueli Maurer, and Dominik Raub. Passive corruption in statistical multi-party computation - (extended abstract). In Adam Smith, editor, *ICITS 12*, volume 7412 of *LNCS*, pages 129–146. Springer, Heidelberg, August 2012. doi:10.1007/978-3-642-32284-6_8.

18 Martin Hirt, Ueli M. Maurer, and Vassilis Zikas. MPC vs. SFE: Unconditional and computational security. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 1–18. Springer, Heidelberg, December 2008. doi:10.1007/978-3-540-89255-7_1.

19 Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 483–500. Springer, Heidelberg, August 2006. doi:10.1007/11818175_29.

20 Jonathan Katz. On achieving the "best of both worlds" in secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 11–20. ACM Press, June 2007. doi:10.1145/1250790.1250793.

21 Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

22 Chen-Da Liu-Zhang, Julian Loss, Ueli Maurer, Tal Moran, and Daniel Tschudi. MPC with synchronous security and asynchronous responsiveness. In *Annual International Conference on the Theory and Application of Cryptology and Information Security - ASIACRYPT 2020*, 2020. to appear.

23 Julian Loss and Tal Moran. Combining asynchronous and synchronous byzantine agreement: The best of both worlds. Cryptology ePrint Archive, Report 2018/235, 2018. URL: `https://eprint.iacr.org/2018/235`.

24 Christoph Lucas, Dominik Raub, and Ueli M. Maurer. Hybrid-secure MPC: trading information-theoretic robustness for computational privacy. In Andréa W. Richa and Rachid Guerraoui, editors, *29th ACM PODC*, pages 219–228. ACM, July 2010. `doi:10.1145/1835698.1835747`.

25 Achour Mostéfaoui, Moumen Hamouma, and Michel Raynal. Signature-free asynchronous Byzantine consensus with $t < n/3$ and $O(n^2)$ messages. In *ACM Symposium on Principles of Distributed Computing*, pages 2–9. ACM, 2014. `doi:10.1145/2611462.2611468`.

26 Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 3–33. Springer, Heidelberg, 2018. `doi:10.1007/978-3-319-78375-8_1`.