Early Stopping for Any Number of Corruptions

Julian Loss[1] and Jesper Buus Nielsen[2]

[1] CISPA Helmholtz Center for Information Security, Germany
[2] Aarhus University, Denmark

Abstract. Minimizing the round complexity of byzantine broadcast is a fundamental question in distributed computing and cryptography. In this work, we present the first *early stopping* byzantine broadcast protocol that tolerates up to $t = n-1$ malicious corruptions and terminates in $\mathcal{O}(\min\{f^2, t+1\})$ rounds for any execution with $f \leq t$ *actual corruptions*. Our protocol is deterministic, adaptively secure, and works assuming a plain public key infrastructure. Prior early-stopping protocols all either require honest majority or tolerate only up to $t = (1-\epsilon)n$ malicious corruptions while requiring either trusted setup or strong number theoretic hardness assumptions. As our key contribution, we show a novel tool called a *polariser* that allows us to transfer certificate-based strategies from the honest majority setting to settings with a dishonest majority.

1 Introduction

In the problem of byzantine broadcast [21], a sender P_s holds a value v that it wants to share among n parties $\mathsf{P}_1, ..., \mathsf{P}_n$ using a distributed protocol Π with the following properties: 1) *validity*: if the sender is honest (i.e., it follows the protocol description of Π correctly), all honest parties output v 2) *agreement*: all honest parties output the same value v' from Π. Broadcast is an integral building block in many cryptographic and and distributed protocols, e.g., multi-party computation, verifiable secret sharing, and state-machine replication. One of the most important efficiency metrics for a broadcast protocol is its *round complexity*: how many rounds does the protocol run for until all parties have terminated?

A seminal result of Dolev and Strong [12] shows that any deterministic broadcast protocol tolerating $t < n$ malicious parties runs for at least $t + 1$ rounds. In the same work, they also give a protocol that shows the tightness of their bound. However, their bound is known to be loose for protocol executions where the number of corruptions f is less than t, i.e., $f < t$. In this case, the tightest lower bound [11] says that any (deterministic) protocol must run for at least $f+2$ rounds. Intrigued by this vexing gap, a long line of work has studied so-called *early stopping* protocols which terminate in $O(g(f)) = o(t)$ rounds (for some function g) in any execution where the actual number of corrupted parties f is sufficiently small compared to t. Early stopping protocols are known for both the information theoretic setting with $t < n/3$ malicious corruptions [6, 17, 2, 22] and the authenticated setting with $t < n/2$ corruptions [27]. To the best of our knowledge, however, little is known about early stopping protocols for the setting of $n/2 \leq t < n$ malicious corruptions. On one hand, several randomized protocols achieve sublinear (in n) round complexity for broadcast in the dishonest majority setting [16, 15, 7, 30, 29]. However, these protocols require the maximum number t of corruptions to be at most a constant fraction of n in order to stop early (some require t to be much smaller).

All in all, for the full corruption threshold $t = n-1$, the tightest lower bound says that any (deterministic) protocol must run for at least $f+2$ rounds, whereas the best protocol uses n rounds in all runs, even for low f. Clearly, there is a fundamental gap in our understanding of early stopping protocols when $t = n-1$. Motivated by this discussion, we pose the following question: *Are there early stopping broadcast protocols tolerating up to $t = n-1$ corruptions?*

1.1 Our Contribution

In this work, we answer this question in the affirmative by providing the first early stopping protocol CDC for arbitrary majority corruption, i.e., $t = n-1$. Concretely, its properties can be summarized as follows:

- CDC tolerates any number $t < n$ of malicious corruptions.
- For any execution with $f \leq \sqrt{t}$ faults, CDC terminates in $\mathcal{O}(f^2)$ rounds. It always terminates within $\mathcal{O}(t)$ rounds. Prior work achieves either $t+1$ rounds deterministically or requires that the maximum number of faults satisfy $t = (1-\epsilon)\cdot n$ for some $\epsilon > 0$ in order to achieve early stopping.
- Our protocol is secure with respect to a *strongly adaptive adversary*. This type of adversary can observe an honest party's messages, corrupt it, and replace its messages with its own before these messages are delivered. This sets our work apart from existing early stopping protocols for the dishonest majority regime with a strongly adaptive adversary. Namely, existing protocols require either 1) strong number theoretic hardness assumptions (i.e., time-locked puzzles) [29, 28] or 2) tolerate only $t = n/2 + O(1)$ malicious corruptions [16, 15] in order to stop early.
- CDC is deterministic and works in the plain public key model. By comparison, existing early stopping protocols for majority corruption are all randomized and, in many cases, require strong setup assumptions.

In summary, CDC is the first early stopping protocol for $t = n-1$ and makes a significant improvement over the state of the art for any execution with $f = o(\sqrt{n})$ corruptions. We give further comparison with existing literature in the related work section.

1.2 Technical Overview

We now give an overview of our techniques. We begin by giving a brief recap of the classical Dolev-Strong protocol [12] DSC which serves as the basis of our construction. We then explain the central difficulty of making this protocol early stopping and our key insights into how to overcome it.

Recap: the Dolev-Strong protocol. DSC is deterministic and achieves a round complexity of $t+1$ against a strongly adaptive malicious adversary corrupting up to $t < n$ parties. Protocol DSC works as follows for a sender P_s holding a message m:

- In round 1, P_s signs m and sends it to all parties.
- In any round $i \leq t$, a party P_j does as follows. If it receives a message m together with a list of valid signatures from j distinct parties for the first time, it adds m to a set of accepted messages $\mathcal{A}$. Then, P_i adds its own signature to the list, and forwards it to all parties so that they add it to $\mathcal{A}$ at most one round later.
- In round $t+1$, a party P_i that receives a message m together with a list of valid signatures from $t+1$ distinct parties for the first time, adds m to $\mathcal{A}$, but does not forward it to all parties in this round (as the protocol ends by the end of the round). Instead P_i, determines its output as follows. If $\mathcal{A} = \{\}$ or $|\mathcal{A}| > 1$, output a default output $\textsc{NoMsg}$. If $\mathcal{A} = \{m\}$, output m.

Clearly, if an honest P_i adds m to $\mathcal{A}$ at any point during the first t rounds of the protocol, all honest parties add it to $\mathcal{A}$ at most one round later. If P_i adds m to $\mathcal{A}$ in round $t+1$, it sees m with $t+1$ signatures and thus knows that at least one honest party P_j has previously signed m in a previous round, namely, round t (otherwise, P_i would have added m to $\mathcal{A}$ in an earlier round). Since P_j was honest in that round, it would have added its own signature to the list of t signatures it needed to add m to $\mathcal{A}$ in its view and passed on the resulting list of $t+1$ signatures to all parties. Hence, all honest parties add m to $\mathcal{A}$ by round $t+1$.

Why making DSC *early stopping is hard.* Stopping DSC early turns out to be very challenging. Surprisingly, however, this does not result from a fully malicious sender P_s that sends conflicting (signed) messages in DSC to break consistency. Namely, note that this behaviour immediately allows all honest parties to detect that P_s is acting dishonestly and prove it to each other. Interestingly, the central difficulty arises from an apparently more benign type of behaviour that does not even require P_s to be malicious, but only crash faulty: P_s can simply not send any message to (some of) the parties. If only the sender is faulty, we would like to stop the protocol immediately rather than having to run it for its entire duration of $t+1$ rounds. But how do honest parties prove to each other that they did not receive a message from P_s? This is a comparatively simple task in the honest majority setting where $t < n/2$. At a high-level, parties can simply collect a certificate of accusations against the sender for not sending them a message. If they can obtain $t+1$ signed accusations, they can disqualify the sender and stop the protocol. On the other hand, if P_s can not be disqualified, then at least one honest party must have received a message from P_s and can forward it to all other parties. Unfortunately, this strategy spectacularly fails when $t \geq n/2$, as it would now become possible for a corrupted coalition of parties to accuse and disqualify an honest sender P_s. Thus, the apparent paradox that we must overcome is to design an analogue of a certificate for the dishonest majority setting.

Polarisers to the rescue. Our key tool for achieving this is a novel primitive called a *polariser*. Informally, a polariser partitions the parties into two 'polarized' sets Alive and Corrupt. Polarisers are updated continuously throughout the protocol and maintain the following properties. First, an honest party P_i accepts a polariser from another party (and subsequently updates its own polariser) only if it itself is in Alive. Moreover, for any party $\mathsf{P} \in$ Corrupt, a polariser contains accusation against P from *all* parties in Alive. And it is ensured that honest parties never accuse honest parties. As it turns out, these properties make it possible for an honest party P_i to justify any decision in our protocol and convince other honest parties to take the same decision. A crucial observation is that it follows from the properties that all honest parties are in Alive and therefore P_i knows it can forward the polariser and have it accepted by other honest parties: they too will find themeselves in Alive. Thus, polarisers act as our drop-in replacement for certificates. The idea behind creating a polariser is surprisingly simple. As an example, suppose the sender P_s does not send a party P_i a message in the first round. Now, P_i can publicly accuse P_s. To deal with having accusations levelled against honest senders, note that honest parties will move P_s to Corrupt only if it was accused by all parties in Alive. Since P_i never accuses an honest sender P_s and is itself in Alive, this precludes honest parties from moving an honest P_s to Corrupt. However, this creates a different problem: if P_j is also dishonest, but is itself in Alive, it can simply not accuse P_s. In this case, P_s cannot be moved from Alive to Corrupt, since not all parties in Alive have accused P_s. To deal with this type of behaviour from dishonest parties, we present a recursive solution for generating a polariser. In our running example from above, we require that either P_j sends whatever it got from P_s or it accuses P_s of cheating. If P_i receives neither of those two things from P_j, P_i knows that P_j is itself corrupt and accordingly accuses P_j publicly. And, importantly, P_i expects every other party P_k to accuse P_j too, or send to P_i the reason that it did not accuse P_j, namely an accusation of P_j against P_s (which would resolve the issue). If P_k does not send an accusation or resolvement, then P_i will accuse P_k, *et cetera*. In this manner, parties recursively accuse each other in a chain of accusations until all corrupted parties were corrupted or the issue at hand resolved. As we show, our algorithm can resolve such a chain of accusations within $\mathcal{O}(f)$ rounds, at which point all parties on the chain (including P_s) can be moved from Alive to Corrupt in the view of every honest party unless a signature from P_s was received by all honest parties.

Justifying outputs and graded broadcast. Using polarisers, it now becomes possible to justify the output (or non-output) of any subprotocol to each other. To do so, parties will either send their entire view of the protocol transcript so far in case a subprotocol produces output, and send a polariser to justify not having output in that subprotocol. We show a protocol GSTM for *graded broadcast* that is inspired by the protocol of Koo et al. [16]. Roughly, GSTM outputs a message m together with a grade $g \in \{0, 1, 2\}$ and a proof π that justifies the combination of m and g. The grade g reflects a party's confidence in its output. Grade $g = 2$ indicates high confidence, meaning that m should be output, and all other parties have grade at least 1 for the same message m. Grade $g = 1$ indicates low confidence in m, meaning that some parties might not have received the message m. Finally, $g = 0$ indicates that the message was not received, the sender was corrupt, and some dummy NoMsg should be output. Validity ensures that all parties output $(m, 2)$ whenever the sender is honest. The crucial property of GSTM, however, is that if a dishonest party can produce a justified output then the grading rules from above apply too. So a corrupted party basically cannot justify an output unless that output could have been produced by an honest party too.

Putting things together: Protocol DC*.* Using GSTM we are able to run a broadcast protocol DC that resembles the well-known phase-king approach [5] from the honest majority setting. In this style of protocol, one rotates through $f + 1$ leaders until agreement on an output is detected. Essentially, each leader is instructed to broadcast via GSTM whatever it received from the previous instance. The crucial idea is that a malicious leader can never undo the progress that the protocol has made so far by making them choose a different output from one that they have already agreed on (or not choosing the output of an honest sender). This is because each time a new broadcast is initiated by a leader, its input must be justified by the entire view of the protocol so far. As discussed above this means that it must use an input which some honest party could have used, or choose to abort the protocol. More precisely, once an output m is received with a positive grade g in the jth leader iteration, parties set $m_j = m$ and broadcast this message once it is their turn to be the leader. (If the previous king aborted then they pick the most recent such value, if there is one, and otherwise they pick a fixed default output). The consistency properties of GSTM and the justification that comes along with any new output ensures that a dishonest leader can never introduce a new message once parties have already agreed on a message m. Once parties see an output m with grade 2, they can detect agreement, forward the justification for this output to all parties, and terminate. After $f + 1$ leader rotations, at least one of the leaders will be honest, at which point the agreement detection is triggered (recall that honest senders always achieve grade 2 by validity of GSTM) and the protocol terminates.

CDC*: Ensuring $O(t)$ round-complexity.* In the worst case, each of the leader iterations in DC identifies only a single party (i.e., the leader) as malicious. Since each of these steps takes roughly

f rounds, we end up with an overall complexity of $O(f^2)$ for DC. To avoid exceeding $O(t)$ for the round complexity, we can simply stop the protocol after running for $\ell = O(t)$ rounds. At this point, we can afford to run an additional instance of DSC to reach agreement. More precisely, any party that has not terminated by round $\ell - 1$ will thus use DSC to broadcast its view of the protocol so far to all parties after completing iteration ℓ. (Running for one more iteration ensures that all parties have time to be forwarded justifications from already terminated parties). Once DSC terminates after another $O(t)$ rounds, all parties can locally decide on a correct output which, by the justification properties of GSTM, will be consistent with parties who have already terminated. This solution, however, still has an undesirable property: honest parties can terminate $O(t)$ rounds apart, whenever one party terminates in iteration $\ell - 1$, but other parties keep on running. This would make the protocol very difficult to use as a subroutine in higher-level application. To have parties terminate one round apart, our actual protocol CDC replaces DSC with a new protocol WES (WES stands for *weak early stopping*) that terminates in $O(f)$ rounds if the sender is honest and $O(t)$ rounds in any other case—furthermore, parties terminate at most one round apart. This allows honest parties who terminate early in DC to always broadcast their justified outputs via WES. If any WES reports a justified output of DC that will be the final output, otherwise the final output will be NoMsg. Parties can terminate immediately upon receiving output from the first terminating instance of WES reporting a justified output of DC. If an honest party saw a justified output of DC it gets reported in $\mathcal{O}(f)$ rounds and the overall protocol terminates in $\mathcal{O}(f^2)$ rounds. If no honest party saw a justified output from DC in $O(t)$ rounds, there are (at least) $f = \sqrt{t}$ corruptions and then the protocol is allowed to run for $\mathcal{O}(t)$ rounds. We believe that WES may have other natural applications.

Achieving Polynomial Complexity. A final wrinkle is that sending along the protocol's entire transcript in every step would result in exponential communication complexity. However, this turns out to be unnecessary, as P_i can simulate another party P_j's behaviour from its past messages. This means that P_j need only send information associated with the most recent protocol step every time it is asked to justify one of its outputs. Thus, our combined protocol CDC ends up with a communication complexity of $O(n^4\lambda)$, where λ denotes the length of a signature.

1.3 Related Work

Below, we summarize some early stopping protocols from the literature.

Deterministic Protocols. To the best of our knowledge, the first early stopping protocol for byzantine agreement (which implies broadcast for $t < n/2$) with optimal resilience $t < n/3$ in the information theoretic setting was due to Berman et al. [6], who gave a protocol with (optimal) round complexity $\min\{f+2, t+1\}$ and exponential communication. Their work builds on earlier work of Berman and Garay [4] who achieved the same round complexity with polynomial complexity and $n > 4t$. Garay and Moses [17, 18] later improved the communication and computation for the corruption-optimal protocol to polynomial. However, their protocol achieves a slightly worse round complexity for the early stopping case of $\min\{f+5, t+1\}$. Much later, Abraham and Dolev [2] gave the first protocol with optimal round complexity $\min\{f+2, t+1\}$ and polynomial communication/computation. In the authenticated setting with $t < n/2$ and plain PKI, Perry and Toueg [27] showed a protocol with polynomial communication and computation complexity and a round complexity of $\min\{2f+4, 2t+2\}$.

Randomized Protocols. In the following, we let δ denote the failure probability of a protocol. There are various randomized protocols for the honest majority setting with constant expected round complexity for both the $t < n/3$ (information theoretic) setting [14, 24] and $t < n/2$ (authenticated) setting [19, 1]. All of the above protocols can be made to terminate early with worst-case failure probability δ by running them for $O(\log(1/\delta))$ iterations (each iteration has constant many rounds).

One can use similar ideas to turn expected constant round protocols in the dishonest majrority setting with $t < n$ corruptions into protocols that always stop early and have failure probability δ. Here, randomized protocols were first explored by Garay et al., who showed an expected $O(2t-n)^2$-round protocol from plain PKI for any $t < n$ [16]. Their approach was later improved by Fitzi and Nielsen [15], who showed a protocol with $O(2t - n)$ complexity in the same setting. These protocols lead to early stopping protocols with round complexities $O(\log(1/\delta) + (2t - n)^2)$ and $O(\log(1/\delta) + 2t - n)$, respectively, and failure probability δ. However, since $O(2t - n) = O(n)$ (regardless of f) whenever $t = (1 - \epsilon) \cdot n$ where $\epsilon > 0$, these protocols are not early stopping.

Chan et al. [7] presented a randomized broadcast protocol with trusted setup and tolerating any $(1 - \epsilon)$-fraction of adaptive corruptions (for an arbitrary constant $\epsilon > 0$) by assuming no after-the-fact removal of messages. Their protocol achieves a round complexity of $O(\log(1/\delta)) \cdot (n/(n-t))$ for failure probability δ. Also assuming trusted setup and no after-the-fact removals, but tolerating up to $t = n - 1$ corruptions, Wan et al. [30] give a protocol achieving expected $O((n/(n - t))^2)$ round complexity and $O(\log(1/\delta)/\log(n/t)) \cdot (n/(n - t))$ complexity for failure probability δ. Protocols tolerating adaptive corruptions with after-the-fact message removals and $t < (1 - \epsilon)n$ corruptions were studied by Wan et al. [30] and more recently by Srinivasan et al. [28], who gave protocols from time-lock-assumptions achieving round complexities of $(n/(n - t))^2 \cdot \mathsf{polylog}(\lambda)$ (for failure probability negligible in λ) and $O(\log(1/\delta)) \cdot (n/(n - t))$, respectively. Most recently, Alexandru et al. [3] showed how to remove the need for trusted setup in order to obtain $O(\log(1/\delta)) \cdot (n/(n - t))$ round complexity.

Although these protocols all achieve early stopping (with failure probability δ) for $t = (1-\epsilon)\cdot n$, they are also not early stopping when $t = n - O(1)$. Namely, in this case, their round complexity is at least $O(n/(n - t)) = O(n)$, regardless of f.

Lower Bounds, communication optimizations, and weaker models. The famous lower bound of Dolev and Strong was later adapted to the early stopping setting in order to show that the best possible round complexity of an early stopping algorithm in an execution with f faults is lower bounded by $\min\{f + 2, t + 1\}$ [11]. The round lower bound for the early stopping setting was later extended by Keidar and Rajsbaum [20] to the setting of omission faults, which can fail to send or receive some of their messages. They demonstrate that early stopping broadcast/agreement algorithms require the same complexity as in the malicious setting if agreement is required to be *uniform*, i.e., omission faulty parties that output, must output consistently with the honest parties. Chandra et al. present reliable broadcast protocols achieving $f + 2$ rounds in the crash fault model and $2f + 3$ rounds in the omission fault model [8]. The latter result was later improved to $f + 2$ by Parvédy and Raynal [26] to $\min\{f + 2, t + 1\}$ round complexity and $O(n^2 \cdot f)$ communication complexity in the omission fault model.

From the perspective of communication complexity, a result by Dolev and Lenzen [9] shows that any (deterministic) early stopping algorithm with optimal round complexity requires sending $O(nt+t^2 f)$ messages. This tightens the famous Dolev-Reischuk bound for the early stopping case [10]. On the other hand, the recent result of Lenzen and Sheikholeslami [22] demonstrate that this bound can be circumvented by presenting an early stopping protocol with (suboptimal) $O(f)$ round complexity but significantly improved $O(nt)$ communication complexity. (We stress that the protocol of Lenzen and Sheikholeslami actually achieves byzantine agreement, i.e., where all parties give input).

2 Preliminaries

We work arborescences Tree, i.e., directed graph trees with a single root and edges pointing towards the leafs. We denote them by trees below. For a tree Tree we use $\mathsf{path} \in \mathsf{Tree}$ to denote a path $(r, \ldots, l)$ from the root r to a leaf l. We let $\mathsf{depth}(\mathsf{Tree}) = \max_{\mathsf{path} \in \mathsf{Tree}} (|\mathsf{path}| - 1)$. The tree with only a root thus has $\mathsf{depth}(\mathsf{Tree}) = 0$ and if the tree is empty $\mathsf{depth}(\mathsf{Tree}) = -1$. For a path $\mathsf{path} = (r, \ldots, l)$ we let $\mathsf{leaf}(\mathsf{path}) = l$. We assume a synchronous model with n parties $\mathbb{P} = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$. The computation proceeds in rounds where in each round each party can send a message to each other parties that is guaranteed to arrive by the end of that round. We assume a rushing adversary that can adaptively corrupt parties and replace any of the messages they sent for a round and which have not yet been delivered with messages of its choice (or simply delete them). We use t to denote the maximum number of corrupted parties and $f \leq t$ to denote the actual number of corrupted parties. We allow t to take any value $0 \leq t \leq n$, though $t = n$ is trivial.

We assume a PKI. In an initial setup round each party P_i generates a key-pair $(\mathsf{sk}_i, \mathsf{vk}_i)$ for a signature scheme and announces vk_i to a public bulletin board. As is standard for this line of work, we assume the Dolev-Yao model [13] and treat signatures as information theoreitc objects with perfect unforgeability. Throughout, we denote the size of a signature in bits as λ.

Definition 1 (Broadcast). *Let Π be a protocol executed by n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$, where a designated* sender P_s *holds input x and each party P_i terminates upon giving output y_i. We say that Π is a* broadcast protocol *if it has the following properties:*

- ***Validity**: If P_s is honest, each honest party P_i outputs $y_i = x$.*
- ***Agreement**: For honest parties P_i and P_j, $y_j = y_i$.*

When describing our protocols we will assume that all parties get input in the same round. If a party has no input in a protocol we assume that they nonetheless get a dummy input, to ensure they know in which round to start running. The dummy input will be tacit. We will also assume that all sub-protocols give outputs in the same round. This ensures that if the parties call a sub-protocol and then proceed when it gives output, then they are still synchronized.

Sequential composition of protocols without simultaneous termination. Under these conditions we will design protocol where parties might terminate at most one round apart. This leads to problems with composition: when using a protocol as a subroutine, we assume parties give outputs in the same round. But in many of our protoocls, parties terminate one round apart. This can, however, be handled using known techniques for sequential composition of protocols without simultaneous termination at a blowup in round complexity of just 2. Details can be found in [23] and Chapter 7 in [25]. Here, we sketch the main idea for completeness. Protocols which assume that the parties start in the same round will be compiled into protocols tolerating that they start one round apart. The compiler works as follows. If parties start a protocol Π one communication round apart, then after P_i sends its messages for protocol round r of Π, it will wait for two underlying communication rounds to ensure it received messages from honest parties sending their messages one underlying communication round later than P_i. Then, P_i computes its messages for the next protocol round and sends them out, waits for two communication rounds *et cetera*. This leaves the problem that the parties might terminate two communication rounds apart. This would be a problem for sequential composition as we want them to start the next protocol only one communication round apart. However, this is easily mitigated by assuming that Π have justified outputs and that parties can use outputs from other parties. When the first party gets an output it then sends it to the other parties. It will arrive within 1 *communication* round. When a party sees a valid incoming protocol output, they adopt this as their own output and forward it to all parties. Now all parties terminate at most one communication round apart. We will call this the *staggering compiler*. To be able to apply it we will ensure that all our protocol have justified outputs which can be forwarded.

3 Polarisers and Transferable Justifiers

We first put in place a tool which will allow us to write later protocols more concisely. The tool is called a *polariser* as it polarises the n parties into two disjoint sets S and T of which we know all honest parties are fully inside one of the sets. An external party might not know whether S or T contain the honest parties, but the honest parties themselves will of course know which set they are in.

Polarisers are used for agreeing on when a corrupted party P_i did *not* send a message. To motivate their design we first discuss this issue. Consider a party P_i which is to send a message m of a particular form to P_k, say it should be signed. We would like to know when this was *not* done and have a *transferable* proof of this.

If we have a bound $t < n/2$ on how many corruptions there can be then this is easy. You can ask P_i to send m via all other parties P_j and have all P_j forward m to P_k or a signature $\sigma_j = \mathsf{Sig}_{\mathsf{sk}_j}((\text{Acc}, \mathsf{P}_i, \mathsf{P}_j))$ which is a signed accusation of P_i that P_j did not send a message. Now P_k either gets m or a certificate of $t+1$ accusations which can act as a transferable proof that P_j did not send a message. In contrast, in the dishonest majority setting, simple majority voting about whether the message was sent will not solve the problem.

Solving this *missing message* problem is a crucial step in our construction. The central tool are polarisers. The core of a polariser will be a tuple $(\mathsf{Alive}, \mathsf{Corrupt}, \mathsf{Accuse})$, where $\mathbb{P} = \mathsf{Alive} \cup \mathsf{Corrupt}$ and for all parties in Alive, we have a signed accusation for each party in $\mathsf{Corrupt}$ that failed to send a message at some point. Assuming that all honest parties send all intended messages and honest parties never make a false accusation, this leaves only two cases when seeing $(\mathsf{Alive}, \mathsf{Corrupt}, \mathsf{Accuse})$. Either all the honest parties are in Alive or all the honest parties are in $\mathsf{Corrupt}$. Namely, if there is an honest party in both Alive and $\mathsf{Corrupt}$ then an honest party claimed an honest party of not having sent some intended message, which does not happen. But it might of course be that all the parties in Alive are corrupted and are falsely accusing the poor honest parties in $\mathsf{Corrupt}$. This is hard to catch in the dishonest majority case where it might be the case that $|\mathsf{Alive}| > |\mathsf{Corrupt}|$ and yet all parties in Alive are corrupt. This prevents external agreement on who is corrupt.

The trick is to give up on externally valid certificates and go for a weaker type of certificate which maintains transferability only within the context of the current protocol. An honest party P_i can check whether $\mathsf{P}_i \in \mathsf{Alive}$ or $\mathsf{P}_i \in \mathsf{Corrupt}$. If $\mathsf{P}_i \in \mathsf{Corrupt}$, then reject the polariser. Note that in this case *all* honest parties are in $\mathsf{Corrupt}$ and will therefore also reject the polariser if it is sent to them. If $\mathsf{P}_i \in \mathsf{Alive}$, then accept the polariser. Note that in this case *all* honest parties are in Alive and will therefore also accept the polariser if it is sent to them.

Definition 2 (Polariser). *Let* $\mathsf{Pol} = (\mathsf{Alive}, \mathsf{Corrupt}, \mathsf{Accuse})$ *be a tuple where we refer to set* Alive *as the* alive parties, *to set* $\mathsf{Corrupt}$ *as the* corrupt parties, *and to set* Accuse *as the* accusations. *We define the following* structural *properties:*

– ***Justifiability.*** *For every* $\mathsf{P}_i \in \mathsf{Corrupt}$ *and for all parties* $\mathsf{P}_j \in \mathsf{Alive}$, *there exists* $\mathsf{A}_{i,j} \in \mathsf{Accuse}$, *where* $\mathsf{A}_{i,j}$ *denotes a value* $(\textsc{Acc}, \mathsf{P}_i, \mathsf{P}_j, \sigma_i)$, *where* $\mathsf{Ver}_{\mathsf{vk}_i}((\textsc{Acc}, \mathsf{P}_i, \mathsf{P}_j), \sigma_i) = \top$.
– ***Completeness.*** $\mathsf{Alive} \cap \mathsf{Corrupt} = \emptyset$ *and* $\mathsf{Alive} \cup \mathsf{Corrupt} = \mathbb{P}$.

We define the following contextual *property:*

– ***Accusation Soundness.*** *If* P_i *and* P_j *are honest then the adversary cannot construct a valid* $\mathsf{A}_{i,j}$ *in PPT, in particular there is no such* $\mathsf{A}_{i,j}$ *in* Accuse.

We call a polariser Pol *a* P_i*-polariser if* $\mathsf{P}_i \in \mathsf{Pol.Corrupt}$. *We use* $\overset{i}{\not\to}$ *to denote the set of* P_i*-polarisers. As with the Landau notation for asymptotic complexity we misuse notation and use* $\mathsf{Pol} = \overset{i}{\not\to}$ *to denote that* $\mathsf{Pol} \in \overset{i}{\not\to}$. *We also sometimes let* $\overset{i}{\not\to}$ *denote a generic element from* $\overset{i}{\not\to}$.

In all our protocols constructing polarisers we only sign messages of the form $(\textsc{Acc}, \mathsf{P}_i, \mathsf{P}_j, \sigma_i)$ if P_j is corrupt. Therefore:

Lemma 1 (Polarisation Lemma). *Let* Honest *be the set of honest parties and let* Pol *be a polariser. Then* $\mathsf{Honest} \subset \mathsf{Pol.Alive}$ *or* $\mathsf{Honest} \subset \mathsf{Pol.Corrupt}$.

Proof. By Completeness, $\mathsf{Honest} \subset \mathsf{Pol.Alive} \cup \mathsf{Pol.Corrupt}$, and it cannot be the case that $\mathsf{P}_i \in \mathsf{Pol.Alive}$ and $\mathsf{P}_j \in \mathsf{Pol.Corrupt}$ are honest, because then by Justifiability $\mathsf{A}_{i,j} \in \mathsf{Accuse}$, contradicting Accusation Soundness. □

3.1 Transferable Justifiers

Our second general tool is the concept of a transferable justifier for a protocol output. Recall that the purpose of polarisers is to get a transferable proof that some party did not send a message. From these, we will build increasingly more complex messages and eventually a justified output. It is helpful to have some general machinery for talking about transferable justifiers.

Definition 3 (Justifier). *We call a PPT predicate* $J : \mathbb{P} \times \{0,1\}^* \times \{0,1\}^* \to \{\top, \bot\}$ *a* justifier predicate. *If for a party* P_i, *a message* m *and a proof* π *we have that* $J(\mathsf{P}_i, m, \pi) = \top$ *then we say that* P_i *accepts the message* m *with justifier* π. *We use* $J(m, \pi) = \top$ *to denote that* $J(\mathsf{P}_i, m, \pi) = \top$ *for all honest* P_i. *We require that justifiers are transferable, i.e., if* P_i *and* P_j *are honest then* $J(\mathsf{P}_i, m, \pi) = \top$ *implies that* $J(\mathsf{P}_j, m, \pi) = \top$. *When we write* $J(m, \pi) = \top$ *then we mean that* $J(\mathsf{P}_i, m, \pi) = \top$ *for all honest parties* P_i. *By transferability this is the implied if it holds for a single honest* P_i.

As a canonical example consider a protocol where P_j was to send a message and let $\textsc{NoMsg}^{(j)}$ be a special symbol denoting that P_j sent no message. Then a justifier predicare for this could be $J(\mathsf{P}_i, \textsc{NoMsg}^{(j)}, \mathsf{Pol}) \equiv \mathsf{P}_i \in \mathsf{Pol.Honest} \wedge \mathsf{Pol} = \overset{j}{\not\to}$, i.e., P_i accepts that P_j sent nothing is Pol proves that P_j is corrupt and if P_i trusts Pol because it say P_i is honest. Note that this justification of $\textsc{NoMsg}^{(j)}$ is transferable qua Lemma 1.

Definition 4 (Justified Inputs). *We say that a protocol Π has* justified inputs *if it takes an input justifier J_{In} as parameter and works for any justifier predicate J_{In}. We write $\Pi_{J_{\mathsf{In}}}$ to specify the value of J_{In} being used in a given run. When a protocol $\Pi_{J_{\mathsf{In}}}$ with justified inputs is being called by an honest party P_i with input x_i then x_i must be of the form $x_i = (m_i, \pi_i)$ such that $J_{\mathsf{In}}(\mathsf{P}_i, m_i, \pi_i) = \top$.*

Definition 5 (Justified Outputs). *We say that a protocol Π has an* output justifier *if the protocol, as part of its code, specifies a justifier predicate J_{Out}. We denote the output justifier of Π by $\Pi.J_{\mathsf{Out}}$. We say that a protocol P_i has* justified outputs *if it has an output justifier and the outputs y_i of honest P_i are of the form $y_i = (m_i, \pi_i)$ and it always holds that $\Pi.J_{\mathsf{Out}}(\mathsf{P}_i, m_i, \pi_i) = \top$ after a run of the protocol with a PPT adversary.*

An important tool in our protocols is that justified outputs can be passed on to other parties. Therefore, not even adversarial parties should be able to claim wrong outputs. We therefore use a notion of an AJO when defining safety properties.

Definition 6 (Adversarial Justified Output (AJO)). *Let Π be a protocol with an output justifier and let $\mathcal{A}$ be a PPT adversary. Consider the following experiment: Run an execution of Π with $\mathcal{A}$ in the role of the adversary. When all honest parties P_i have produced an output $y_i = (m_i, \pi_i)$, give all y_i to $\mathcal{A}$. We say that $\mathcal{A}$ generates ℓ* adversarial justified outputs (AJOs) *$(m^1, \ldots, m^\ell)$ if it outputs $(\mathsf{P}^1, m^1, \pi^1), \ldots, (\mathsf{P}^\ell, m^\ell, \pi^\ell)$ such that for all $j = 1, \ldots, \ell$ such that P^j is honest and $\Pi.J_{\mathsf{Out}}(\mathsf{P}^j, m^j, \pi^j) = \top$. Otherwise, we say that no outputs were generated.*

Note that the triple $(\mathsf{P}^j, m^j, \pi^j)$ with $\Pi.J_{\mathsf{Out}}(\mathsf{P}^j, m^j, \pi^j) = \top$ does not mean that P^j *produced* the output (m^j, π^j). It merely means that P^j would *accept* the output (m^j, π^j) given its current state and the predicate $\Pi.J_{\mathsf{Out}}$. Note also that if a property holds for all AJOs it also holds for honest outputs as the adversary are given the honest outputs and can reuse them as a triple in $(\mathsf{P}^1, m^1, \pi^1), \ldots, (\mathsf{P}^\ell, m^\ell, \pi^\ell)$.

4 Send Transferable Messages

We now present a protocol which forces a potentially corrupt sender to send a message which the receiver can later transfer. This solves the *missing message problem* discussed earlier. Throughput, we let NoMsg be a designated symbol where NoMsg $\notin \{0,1\}^*$. We use it to signal that a sender was corrupt and did not send a normal message. Ultimately, NoMsg could be mapped to a normal message, like the empty string, but it helps the exposition to assume NoMsg $\notin \{0,1\}^*$. We also use another such symbol $\perp$ and assume that $\perp \notin \{0,1\}^*$ and $\perp \neq$ NoMsg.

Definition 7 (Send Transferable Message Protocol). *Let $\Pi_{J_{\mathsf{Msg}}}$ be a protocol run among n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ where J_{Msg} is the parametrisable input justifier predicate. Assume that $\Pi_{J_{\mathsf{Msg}}}$ specifies a designated sender P_s holding an input $m \in \{0,1\}^*$ along with a justifier π such that $J_{\mathsf{Msg}}(\mathsf{P}_s, m, \pi) = \top$ and parties terminate upon generating output y_i in P_i. The protocol specifies an output justifier predicate $\Pi.J_{\mathsf{Out}}$.*

- ***Correctness:*** *Honest P_i outputs $y_i = (m_i, \pi_i)$, where $m_i \in \{0,1\}^* \cup \{$NoMsg$\}$ and $\pi_i \in \{0,1\}^*$.*
- ***Justified Output:*** Outputs are justified. *When honest P_i outputs $y_i = (m_i, \pi_i)$ then $J_{\mathsf{Out}}(\mathsf{P}_i, m_i, \pi_i) = \top$.*

- ***Justified Message:*** Only justified inputs can appear in justified outputs. *For all AJOs* $y_i = (m_i, \pi_i)$ *either* $m_i = \text{NoMsg}$ *or the justifier* π_i *is of the form* $\pi_i = (\pi_i^1, \pi_i^2)$ *and* $J_{\mathsf{Msg}}(\mathsf{P}_i, m_i, \pi_i^1) = \top$ *for all honest* P_i.
- ***Validity:*** Honest senders manage to send their intended message and only that message. *If* P_s *is honest and has input* (m, π)*, then all honest parties* P_j *have output* $y_j = (m_j, \pi_j)$ *with* $m_j = m$*. Furthermore, for all AJOs* m' *it holds that* $m' = m$.
- ***Agreement:*** All outputs are the same or NoMsg. *For all AJOs* m^1 *and* m^2 *it holds that* $m^1 = \text{NoMsg}$ *or* $m^2 = \text{NoMsg}$ *or* $m^1 = m^2$.

We say that Π *is a* send transferable message with agreement (STMA) protocol with J_{Out}-justified output *if it has the above properties. If it lacks agreement we call it an STM protocol. If an STMA protocol has the additional property that* $m \neq \text{NoMsg}$ *for all AJOs, then we call it a* justifiable broadcast*. We call an output a* legal *STM output if it has the correctness and justified output properties.*

Remark 1. Note that justifiable broadcast ensures that all outputs are the same, so it implies the notion of broadcast in Definition 1. It additionally has input and output justifiers, which will be convenient when using it as sub-protocol. It is straight forward to see that we can create a justified broadcast protocol $\mathsf{DSC}_{\mathsf{P}_s, J_{\mathsf{Msg}}}$ by using Dolev-Strong with sender P_s and $t = n-1$ corruptions and where parties only accept signatures from P_s on (m, π_{MSG}) where $J_{\mathsf{Msg}}(m, \pi_{\text{MSG}}) = \top$. The round complexity is $\mathcal{O}(n)$. We use this protocol later.

Remark 2. Note that honest parties have input $m \neq \text{NoMsg}$, so by Validity, if output $m = \text{NoMsg}$ can be justified, then P_s is corrupt.

Remark 3. We have required that for all AJOs $y_i = (m_i, \pi_i)$ either $m_i = \text{NoMsg}$ or the justifier π_i is of the form $\pi_i = (\pi_i^1, \pi_i^2)$ and $J_{\mathsf{Msg}}(\mathsf{P}_i, m_i, \pi_i^1) = \top$ for all honest P_i. Notice that the definition does not restrict what π_i^2 is and how it affects whether J_{Out} accepts an output. Typically π_i^2 will be a protocol dependent value proving that the message m_i resulted from running Π and J_{Out} will check that this is the case. Presumably π_i^2 will also contain a signature m_i from P_s to ensure validity.

4.1 Polariser Cast

We present a STM protocol PC called *polariser cast*. The protocol will proceed roughly as follows.

1. The sender signs its justified input and sends it to all parties.
2. If the sender P_s did not send a justified signed input in round 0 then each P_j will accuse P_s.
3. If P_s should have been accused but a party P_j did not accuse P_s in round 1, then all parties P_k will accuse P_j in round 2, unless $\mathsf{P}_j = \mathsf{P}_s$ such that it was already accused, *et cetera*.
4. Since only corrupted parties are accused, at some point there are no more parties to accuse, and at this point an output can be computed. Either P_s gave an input or P_s can be moved to $\mathsf{Corrupt}$ along with all parties with enough accusations.
5. This gives each party an output candidate, but different honest parties might hold different messages m.
6. The parties exchange their output candidates, and if some P_j has different signed values in any of them, then this is used as a transferable proof that P_s is corrupt.

Before describing the protocol in detail we give some helping definitions. During the protocol each P_i will keep a set S of values received. We refer to these values as *elements*.

Definition 8 (Well-formed elements). *We call an element* e well-formed *if it is of one of the following two forms.*

Inputs: $e = (\textsc{In}, m, \pi, \sigma)$*, where* $\mathsf{Ver}_{\mathsf{vk}_s}((\textsc{In}, m), \sigma) = \top$ *and* $P(\mathsf{P}_s, m, \pi) = \top$*.*
Accusations: $e = (\textsc{Acc}, \mathsf{P}_i, \mathsf{P}_j, \sigma_i)$*, where* $\mathsf{Ver}_{\mathsf{vk}_i}((\textsc{Acc}, \mathsf{P}_i, \mathsf{P}_j), \sigma_i) = \top$*.*

Each P_i has its own version of S. When we need to distinguish it we denote it by S_i. We use S_i^r to denote the value of S_i at P_i in round r. We define some helper functions used in the protocol.

Detect corruption: The function $\mathsf{ToAccuse}^r(S) \subset \mathbb{P}$ takes as input a set of well-formed elements and a round number r and computes a set of parties $\mathcal{P} \in \mathbb{P}$, which we think of as being corrupt.
Complete: We call a set of well-formed elements S *complete* if there are no more parties to accuse. More precisely, $\mathsf{Complete}(S) \equiv \exists r > 0\, (\mathsf{ToAccuse}^r(S) = \emptyset)$.
Output: The function $\mathsf{Out}(S)$ takes as input a complete set of well-formed elements and computes a possible output of PC, i.e., if $\mathsf{Complete}(S)$ then $\mathsf{Out}(S) = (m, \pi)$ where π is a signature on m under vk_s or $\mathsf{Out}(S) = (\textsc{NoMsg}, \mathsf{Pol})$ where $\mathsf{Pol} = \overset{s}{\not\to}$.

$\mathsf{PC}(\mathsf{P}_s, m, \pi)$

Input: In round 0 all P_i initialise $y_i = \bot$, $S_i = \emptyset$ and the sender P_s computes $\sigma = \mathsf{Sig}_{\mathsf{sk}_s}((\textsc{In}, m))$, sends $e = (\textsc{In}, m, \pi, \sigma)$ to all parties and adds it to S_s.
Basic Loop: For $r = 1, \dots$ party P_i does the following in round r:

1. Receive well-formed elements e sent in round $r-1$ and add them to S_i if the following holds. If e is an input, then only add e to S if there is not already an input in S_i. If e is an accusation of $(\textsc{Acc}, \mathsf{P}_i, \mathsf{P}_j, \sigma_i)$ then only add it if there is not already some $(\textsc{Acc}, \mathsf{P}_i, \mathsf{P}_j, \cdot)$ in S_i.
2. Compute $\mathcal{P}_i = \mathsf{ToAccuse}^r(S_i)$ and for each $\mathsf{P}_j \in \mathcal{P}_i$ add $e = (\textsc{Acc}, \mathsf{P}_i, \mathsf{P}_j, \mathsf{Sig}_{\mathsf{sk}_i}((\textsc{Acc}, \mathsf{P}_i, \mathsf{P}_j)))$ to S_i.
3. *Echo rule*: If e was added to S_i above send it to all parties in this round.

Produce Candidate: In the first round r where it happens that $\mathsf{Complete}(S_i)$ and $y_i = \bot$, compute $y_i = \mathsf{Out}(S_i)$ and send y_i to all other parties.
Adopt Candidate: In the first round r where it happens that $y_i = \bot$ and some P_j sent y_j with $J_{\mathsf{Out}}(\mathsf{P}_i, y_j) = \top$ let $y_i = y_j$ and send y_i to all parties.
Termination: In the first round r where it happens that $y_i \neq \bot$, run for one more round and then terminate with output y_i.
Output Justification: We define $\mathsf{PC}.J_{\mathsf{Out}}(\mathsf{P}_i, y_i)$. Parse $y_i = (m, \pi)$ and check that *either* $y_i = (\textsc{NoMsg}, \mathsf{Pol})$ and $\mathsf{P}_i \in \mathsf{Pol.Honest}$ and $\mathsf{Pol} = \overset{s}{\not\to}$ *or* $m \neq \textsc{NoMsg}$ and $\pi = (\pi^1, \pi^2)$ and $J_{\mathsf{In}}(m, \pi^1) = \top$ and $\mathsf{Ver}_{\mathsf{vk}_s}(m, \pi^2) = \top$.

Fig. 1. A Sending Transferable Message Protocol.

The protocol is given in Fig. 1. We now proceed to define $\mathsf{ToAccuse}$ and Out. We use a tree-based definition where Tree is a tree of missing accusations. What messages are missing depends on what round we are in, so the function Tree^r also depends on the round number r. We write $\mathsf{Tree}^r(S)$ to denote that the function Tree^r is applied to the set S. The nodes of the tree will be elements $(\mathsf{P}, \rho) \in \mathbb{P} \times \mathbb{N}$.

Definition 9 (Tree Function). *The function* $\mathsf{Tree}^r(S)$ *is defined as follows:*

1. *Let* T *be the empty tree.*

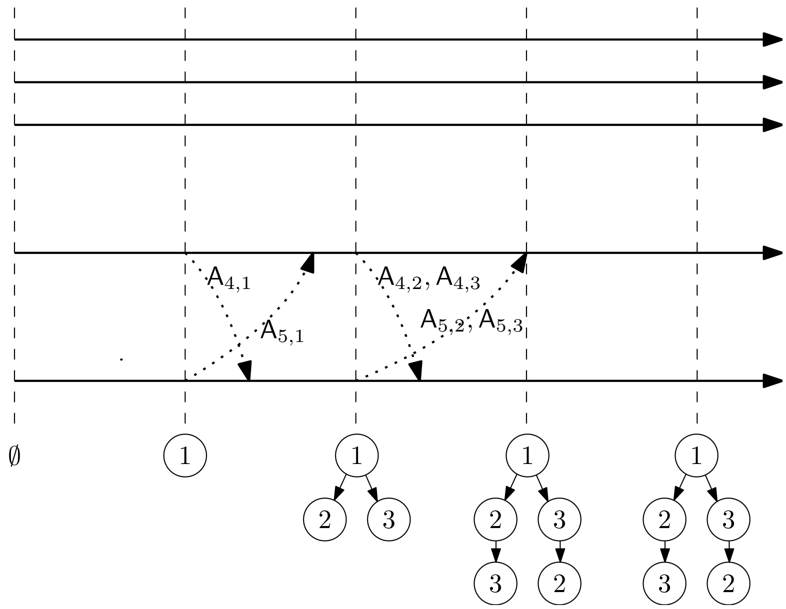

Fig. 2. Party P_1 is the sender. Parties P_1, P_2, and P_3 are corrupted. Their timelines are shown as the top three. Parties P_4 and P_5 are honest and their timelines shown at the bottom. Time runs from left to right and vertical dashed lines are round separators with the first one showing the beginning of round 0. To not clutter the figure, we do not show messages sent *to* corrupted parties. Below the timelines we show the tree built by P_5. In round 0 it is empty. In round 0 party P_1 does not send its signature σ. Therefore P_5 adds the root $(\mathsf{P}_1, 0)$. We are then in round 1, so all parties in leafs of paths of length 1 should be accused, i.e., P_1. In round 1 party P_4 therefore accuses P_1 and P_5 also accuses P_1. The corrupted parties do not accuse P_1. Therefore P_5 adds edges from $(\mathsf{P}_1, 0)$ to $(\mathsf{P}_2, 1)$ and $(\mathsf{P}_3, 1)$. We are then in round 2, so all parties in leafs of paths of length 2 should be accused, so both honest parties accuse P_2 and P_3. The corrupted parties do not accuse. Therefore P_3's missing accusation of P_2 is added as an edge and P_2's missing accusation of P_3 is added as an edge. Note that for instance P_1's missing accusation of P_2 is not added as an edge as we only add parties not already on a path. We are then in round 3 so parties in leafs on paths of length 3 should be accused, i.e., P_3 and P_2. However, no accusations are actually sent as equivalent accusations were sent already. We are then in round 4. No new nodes are added. Parties in leafs of paths of length 4 should be accused. There are no such paths, so the accusation is considered complete, and the protocol ends. We have $\mathsf{Alive} = \{\mathsf{P}_4, \mathsf{P}_5\}$, $\mathsf{Corrupt} = \{\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3\}$, and $\mathsf{Accuse} = \{\mathsf{A}_{4,1}, \mathsf{A}_{5,1}, \mathsf{A}_{4,2}, \mathsf{A}_{4,3}, \mathsf{A}_{5,2}, \mathsf{A}_{5,3}\}$, so we have a legal P_1-polariser.

2. *If* $\nexists(\textsc{In}, m, \cdot, \cdot) \in S$ *then add* $(\mathsf{P}_s, 0)$ *to* T *as the root.*
3. *For* $\rho = 1, \ldots, r$*:*
   (a) *For all* $\mathsf{path} \in T$ *with* $|\mathsf{path}| = \rho$*:*
      i. *Let* $(\mathsf{P}_j, \rho - 1) = \mathsf{leaf}(\mathsf{path})$.
      ii. *For all* $\mathsf{P}_k \in \mathbb{P}$ *where* $(\mathsf{P}_k, \cdot) \notin \mathsf{path}$ *and* $\nexists(\textsc{Acc}, \mathsf{P}_k, \mathsf{P}_j, \cdot) \in S$*, add the edge* $((\mathsf{P}_j, \rho - 1), (\mathsf{P}_k, \rho))$ *to* T.
4. *Output* T.

We think of $\mathsf{Tree}^r(S)$ as the tree of missing elements relative to an honest run of $\mathsf{PC}(\mathsf{P}_s, \ldots)$. As an example, if P_s is honest it should send $(\textsc{In}, m, \cdot, \cdot)$ in round 0, so if $\nexists(\textsc{In}, m, \cdot, \cdot) \in S$ then we add $(\mathsf{P}_s, 0)$ to signify that P_s omitted a message in round 0. Therefore, if the tree has the path $((\mathsf{P}_s, 0))$ then all P_k should have accused P_s in round 1. If P_k does not do this, then in the iteration of the loop with $\rho = 1$ the path $\mathsf{path} = ((\mathsf{P}_s, 0))$ of length 1 will get considered and so will $(\mathsf{P}_s, 0) = \mathsf{leaf}(\mathsf{path})$. So if $\mathsf{P}_k \neq \mathsf{P}_s$ did not accuse P_s then we will have $\nexists(\textsc{Acc}, \mathsf{P}_k, \mathsf{P}_s, \cdot) \in S$ and hence $((\mathsf{P}_s, 0), (\mathsf{P}_k, 1))$ gets added to the tree T. So we add an edge to $(\mathsf{P}_k, 1)$ to signify that in round 1 party P_k failed an obligation and then points from $(\mathsf{P}_s, 0)$ to say that the obligation was to accuse P_s because P_s failed an obligation in the previous round. The reason that Tree^r takes r as input is that some accusation might be missing simply because the parties did not have a chance to send them yet.

We now define $\mathsf{ToAccuse}^r(S)$. To motivate the definition, consider a tree $\mathsf{Tree}^{r-1}(S_i)$ at P_i at the beginning of round r, i.e., after receiving the accusations $(\textsc{Acc}, \cdot, \cdot)$ sent out in round $r - 1$. If $\mathsf{path} = (\ldots, (\mathsf{P}_j, r - 1)) \in \mathsf{Tree}^{r-1}(S_i)$ this is because P_j missed an obligation in round $r - 1$. Therefore P_i must accuse P_j. This motivate the following definition

$$\mathsf{ToAccuse}^r(S) = \left\{ \mathsf{P}_j \,\middle|\, \exists(\ldots, (\mathsf{P}_j, r - 1)) \in \mathsf{Tree}^{r-1}(S) \right\} \ .$$

If a set S is complete then it allows to compute an output as follows.

Definition 10 (Output). *The function* $\mathsf{Out}(S)$ *is defined as follows.*

1. *The input is a complete set* S*, so we can find the smallest* r *such that* $\mathsf{ToAccuse}^r(S) = \emptyset$.
2. *If* $r = 1$ *then pick* $(\textsc{In}, m, \sigma, \pi) \in S$ *and output* $(m, (\sigma, \pi))$.
3. *If* $r > 1$ *then output* $(\textsc{NoMsg}, \mathsf{Pol} = (\mathsf{Alive}, \mathsf{Corrupt}, \mathsf{Accuse}))$*, where*

$$\begin{aligned} \mathsf{Corrupt} &= \{ \mathsf{P}_j \mid \exists(\mathsf{P}_j, \cdot) \in \mathsf{nodes}(\mathsf{Tree}^r(S)) \} \ , \\ \mathsf{Accuse} &= S \ , \\ \mathsf{Alive} &= \mathbb{P} \setminus \mathsf{Corrupt} \ . \end{aligned}$$

We proceed to prove PC secure when using the above definitions of $\mathsf{ToAccuse}$, $\mathsf{Complete}$, and Out. Before the proof it may be instructive to consider the example runs of the protocol in Figs. 2 to 4.

Definition 11 (equivalent sets). *Let* S *and* T *be two sets of well-formed elements. We say that* $S \sqsubseteq T$ *if* $\exists(\textsc{In}, m, \pi, \sigma_s) \in S$ *implies that* $\exists(\textsc{In}, m', \pi', \sigma'_s) \in T$ *and* $\exists(\textsc{Acc}, \mathsf{P}_i, \mathsf{P}_j, \sigma_i) \in S$ *implies that* $\exists(\textsc{Acc}, \mathsf{P}_i, \mathsf{P}_j, \sigma'_i) \in T$*. We call two sets equivalent if* $S \sqsubseteq T$ *and* $T \sqsubseteq S$*. We call two elements equivalent if* $\{e_1\}$ *and* $\{e_2\}$ *are equivalent.*

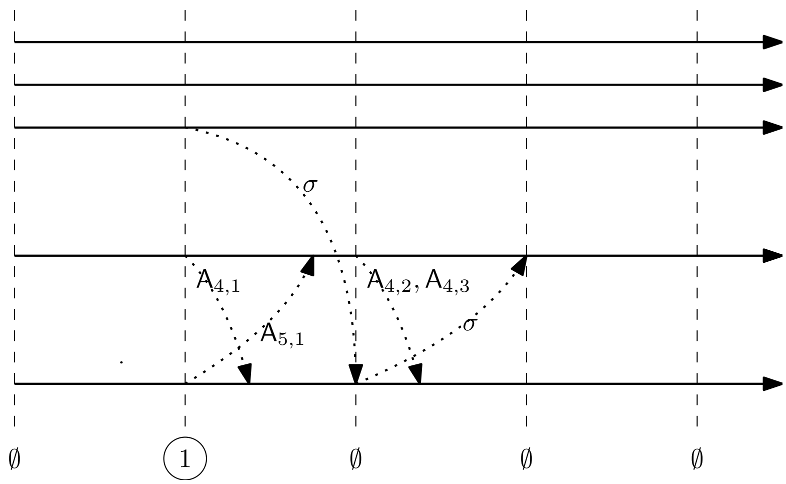

Fig. 3. For explanation notation see Fig. 2. In round 0 party P_1 does not send its signature σ. Therefore P_5 adds the root $(\mathsf{P}_1, 0)$. We are then in round 1 and parties P_4 and P_5 accuse P_1. The corrupted parties do not accuse P_1, but P_3 forwards a signature σ by P_1 to P_5. Therefore the tree computed by P_5 in round 2 is again empty. Therefore the accusation is over for P_5. It terminates with output σ. Note that P_4 is in the same situation as in Fig. 2. Its tree will look like that of P_5 in round 2 in Fig. 2. So, it accuses P_2 and P_3. By the echo rule P_5 forwards σ to P_4 which will then have an empty tree by round 3 and terminate with output σ.

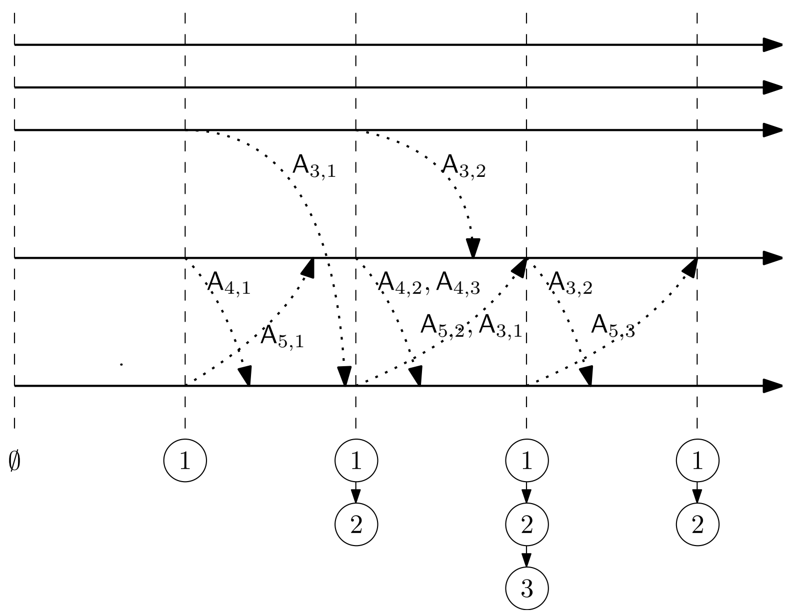

Fig. 4. For explanation notation see Fig. 2. In round 0 party P_1 does not send its signature σ. Therefore P_5 adds the root $(\mathsf{P}_1, 0)$. We are then in round 1 and parties P_4 and P_5 accuse P_1. Party P_3 accuses P_1 but sends the accusation only to P_5. Party P_4 does not accuse P_1. Party P_5 adds an adge representing the missing accusation of P_1 by P_2. Party P_4 being in the same situation as in Fig. 2 accuses P_2 and P_3. Party P_5 accuses all parties which are leafs on paths of lenght 2, i.e., party P_2. It also forwards $\mathsf{A}_{3,1}$ because of the echo rule. Now P_3 acuses P_2 but only towards P_4. Therefore P_5 is missing the accusation of P_2 by P_3 and adds an edge to represent it. It this has a path of lenght 3 in round 3 and thus accuses P_3. But in the same round P_4 forwards $\mathsf{A}_{3,2}$ because of the echo rule. Therefore by round 4 the tree computed by party P_5 is back to height 1 and the protocol ends for P_5. It outputs $\mathsf{Alive} = \{\mathsf{P}_3, \mathsf{P}_4, \mathsf{P}_5\}$, $\mathsf{Corrupt} = \{\mathsf{P}_1, \mathsf{P}_2\}$ and $\mathsf{Accuse} = \{\mathsf{A}_{4,1}, \mathsf{A}_{5,1}, \mathsf{A}_{3,1}, \mathsf{A}_{5,2}, \mathsf{A}_{4,2}, \mathsf{A}_{5,3}, \mathsf{A}_{3,2}\}$, which is a legal P_1-polariser.

Lemma 2 (propagation lemma). *For all honest* P_i *and* P_j *and rounds* $r > 0$ *reached in the protocol it holds that* $S_i^{r-1} \sqsubseteq S_j^r$.

Proof. This follows from the fact that all elements s added to S_i gets forwarded to all P_j and if s is considered well-formed by P_i then it is also considered well-formed by P_j. Therefore it is added to S_j^r (unless there already was an equivalent element in S_j^r). □

Lemma 3 (tree monotonicity lemma). *For all sets of well formed elements* S, T *and all* $r \geq 0$ *its holds that*

$$S \sqsubseteq T \Longrightarrow \mathsf{Tree}^r(T) \subseteq \mathsf{Tree}^r(S) \ .$$

Proof. Note that for an object o, root or edge, to be included in $\mathsf{Tree}^r(T)$ it is required that some element is missing in T, i.e., $\nexists(\text{IN}, m, \cdot, \cdot) \in T$ or $\nexists(\text{ACC}, \mathsf{P}_k, \mathsf{P}_j, \cdot) \in T$. Since $S \sqsubseteq T$ these conditions respectively imply that $\nexists(\text{IN}, m, \cdot, \cdot) \in S$ and $\nexists(\text{ACC}, \mathsf{P}_k, \mathsf{P}_j, \cdot) \in S$, so the same object o gets included in $\mathsf{Tree}^r(S)$. □

The following corollary is important in showing that no honest parties are accused by honest parties. The tree $\mathsf{Tree}^{r-1}(S_i^{r-1})$ is the tree that P_i used in round $r-1$ to calculate who it should accuse. The tree $\mathsf{Tree}^{r-1}(S_j^r)$ is the tree that S_j uses in round r to calculate who S_i ought to have sent an accusation against—it uses the same function Tree^{r-1}, but its own current set S_j^r. If $\mathsf{Tree}^{r-1}(S_j^r) \sqsubseteq \mathsf{Tree}^{r-1}(S_i^{r-1})$ then P_j does not expect to receive any accusations which were not sent.

Corollary 1. *For all honest* P_i *and* P_j *and rounds* $r > 0$ *reached in the protocol it holds that*

$$\mathsf{Tree}^{r-1}(S_j^r) \sqsubseteq \mathsf{Tree}^{r-1}(S_i^{r-1}) \ .$$

Proof. By the two preceding lemmas we have that $S_i^{r-1} \sqsubseteq S_j^r$ and that $S \sqsubseteq T \Longrightarrow \mathsf{Tree}^{\rho}(T) \subseteq \mathsf{Tree}^{\rho}(S)$ for all S, T and ρ. Set $S = S_i^{r-1}$, $T = S_j^r$ and $\rho = r-1$. □

Lemma 4. *If* P_i *is honest and accuses* P_j *then* P_j *is corrupted.*

Proof. By construction, if P_i accuses P_j in round r then $\mathsf{P}_j \in \mathsf{ToAccuse}^r(S_i^r)$. By definition this means that $\exists(\ldots, (\mathsf{P}_j, r-1)) \in \mathsf{Tree}^{r-1}(S_i^r)$. If $r = 1$ then this implies that $\exists((\mathsf{P}_j, 0)) \in \mathsf{Tree}^0(S_i^r)$, and hence $\mathsf{P}_j = \mathsf{P}_s$, as only P_s can occur in the root. Therefore $\nexists(\text{IN}, m, \cdot, \cdot) \in S_i$. Hence, $\mathsf{P}_j = \mathsf{P}_s$ did not send its signed input to P_i in round 0. Therefore $\mathsf{P}_j = \mathsf{P}_s$ is corrupted. If $r > 1$ then we have that $\exists(\ldots, (\mathsf{P}_k, r-2), (\mathsf{P}_j, r-1)) \in \mathsf{Tree}^{r-1}(S_i^r)$. By construction (see Item 3(a)ii in Definition 9) this implies that $\exists(\ldots, (\mathsf{P}_k, r-2)) \in \mathsf{Tree}^{r-1}(S_i^r)$ and $\nexists(\text{ACC}, \mathsf{P}_j, \mathsf{P}_k, \cdot) \in S_i^r$. If P_j is honest this implies that $\exists(\ldots, (\mathsf{P}_k, r-2)) \in \mathsf{Tree}^{r-1}(S_j^{r-1})$, as $\mathsf{Tree}^{r-1}(S_i^r) \sqsubseteq \mathsf{Tree}^{r-1}(S_j^{r-1})$. Therefore $\mathsf{P}_k \in \mathsf{ToAccuse}^{r-1}(S_j^{r-1})$. So if P_j is honest it sent $(\text{ACC}, \mathsf{P}_j, \mathsf{P}_k, \cdot)$ to P_i in round $r-1$. This contradicts $\nexists(\text{ACC}, \mathsf{P}_j, \mathsf{P}_k, \cdot) \in S_i^r$. □

The following lemma shows that when the set S is complete at an honest party such that it terminates, then $\mathsf{Out}(S)$ produces a correct output. In particular, if there is no signature in S, then a polariser is produced proving that P_s is corrupt.

Lemma 5 (justified output). *If* S *is a set held by an honest party* P_i *and* $\mathsf{Complete}(S)$*, such that* P_i *produces output* $\mathsf{Out}(S)$*, then* $\mathsf{PC}.J_{\mathsf{Out}}(\mathsf{P}_i, \mathsf{Out}(S)) = \top$.

Proof. We want to prove that $\mathsf{PC}.J_{\mathsf{Out}}(\mathsf{P}_i, \mathsf{Out}(S)) = \top$. This means that if we let $y_i = \mathsf{Out}(S)$ as defined in Definition 10 then we have to make sure that $\mathsf{PC}.J_{\mathsf{Out}}(\mathsf{P}_i, y_i) = \top$ where $\mathsf{PC}.J_{\mathsf{Out}}$ is defined in **Output Justification** in Fig. 1. We write this out. First parse $y_i = (m, \pi)$. We then have to prove that *either* 1) $y_i = (\textsc{NoMsg}, \mathsf{Pol})$ and $\mathsf{P}_i \in \mathsf{Pol.Honest}$ and $\mathsf{Pol} = \overset{s}{\not\to}$ *or* 2) $m \neq \textsc{NoMsg}$ and $\pi = (\pi^1, \pi^2)$ and $J_{\mathsf{In}}(m, \pi^1) = \top$ and $\mathsf{Ver}_{\mathsf{vk}_s}(m, \pi^2) = \top$.

By $\mathsf{Complete}(S)$ there exists r such that $\mathsf{ToAccuse}^r(S) = \emptyset$. Assume that $r = 1$. From $\mathsf{Complete}(S)$ we get that $\mathsf{ToAccuse}^1(S) = \emptyset$. This implies that $\nexists(\ldots, (\mathsf{P}_j, r-1)) \in \mathsf{Tree}^{r-1}(S)$ which for $r = 1$ means that $\nexists((\mathsf{P}_j, 0)) \in \mathsf{Tree}^0(S)$, which by the construction of the tree T in the algorithm Tree^0 defined in Definition 9 means that it is *not* the case that $\nexists(\textsc{In}, m, \cdot, \cdot) \in S$. So, there is some well-formed $(\textsc{In}, m, \cdot, \cdot) \in S$. Therefore the output is of the form in case 2 above.

Assume then that $r > 1$. From $\mathsf{Complete}(S)$ we get that $\mathsf{ToAccuse}^r(S) = \emptyset$ and by r being minimal we have that $\mathsf{ToAccuse}^{r-1}(S) \neq \emptyset$. From $\mathsf{ToAccuse}^{r-1}(S) \neq \emptyset$, it follows that there is at least one path in $\mathsf{Tree}^{r-1}(S)$ and hence also a root. Therefore $(\mathsf{P}_s, 0) \in \mathsf{nodes}(\mathsf{Tree}^{r-1}(S))$ and hence $\mathsf{P}_s \in \mathsf{Pol.Corrupt}$. We therefore just have to show that Pol is a legal polariser. Completeness follows from $\mathsf{Alive} = \mathbb{P} \setminus \mathsf{Corrupt}$. Since S contains only well-formed elements and $\mathsf{Accuse} = S$ by Definition 10, for justifiability it is sufficient to prove that for all $(\mathsf{P}_i, \mathsf{P}_j) \in \mathsf{Alive} \times \mathsf{Corrupt}$ it holds that $(\mathsf{Accuse}, \mathsf{P}_i, \mathsf{P}_j, \cdot) \in S$. So, assume that $(\mathsf{P}_i, \mathsf{P}_j) \in \mathsf{Alive} \times \mathsf{Corrupt}$. This implies that $(\mathsf{P}_j, \rho) \in \mathsf{nodes}(\mathsf{Tree}^{\rho}(S))$ for some $\rho < r$. We argue that this implies that S contains $(\textsc{Acc}, \mathsf{P}_i, \mathsf{P}_j, \cdot)$. Assume to the contrary that S does not contain $(\textsc{Acc}, \mathsf{P}_i, \mathsf{P}_j, \cdot)$. Then it would be the case that $((\mathsf{P}_j, \rho), (\mathsf{P}_i, \rho+1)) \in \mathsf{Tree}^r(S)$ by Item 3(a)ii in Definition 9 unless $(\mathsf{P}_i, \cdot)$ was already in the path in question (but $(\mathsf{P}_i, \cdot)$ being on the path contradicts $\mathsf{P}_i \in \mathsf{Alive}$ as we added to $\mathsf{Corrupt}$ all parties on all paths by construction of Definition 10). But if $((\mathsf{P}_j, \rho), (\mathsf{P}_i, \rho+1)) \in \mathsf{Tree}^r(S)$ then from $\rho+1 \leq r$ and because we assume that $(\textsc{Acc}, \mathsf{P}_i, \mathsf{P}_j, \cdot) \notin S$ it is not the case that $\mathsf{ToAccuse}^r(S) \neq \emptyset$, as we would have $\mathsf{P}_i \in \mathsf{ToAccuse}^r(S)$ by construction. Since we have as premise that $\mathsf{ToAccuse}^r(S) = \emptyset$ it follows that $(\textsc{Acc}, \mathsf{P}_i, \mathsf{P}_j, \cdot) \in S$, as desired. □

Theorem 1. *The protocol* PC *is an STM protocol for* $t < n$. *Furthermore, assume that* PC *has inputs* (m, π) *with* $|(m, \pi)| \leq \ell$. *Let* λ *be the length of a signature. Then the protocol has communication complexity* $\mathcal{O}(n^2\ell + n^4\lambda)$ *and the size of the justified output is at most* $\mathcal{O}(\ell + n^2\lambda)$. *The protocol uses at most* $f + 1$ *rounds.*

Proof. Correctness follows by construction of Out. Justified output follows from Lemma 5. Justified message follows by construction of J_{Out}. Validity follows by the fact that if P_s is honest then in any polariser Pol accepted by an honest party it holds that $\mathsf{P}_s \in \mathsf{Alive}$ by Lemma 4. We then count communicaiton complexity. We ignore constant factors in the counting. In round 0 party P_s sends to all parties its input of lenght ℓ and a signature. This is the sending of $n(\ell + \lambda)$ bits. During the basic loop each party forwards at most one well-formed input from P_s to other parties, so this is at most $n^2(\ell + \lambda)$ bits. Besides this, each P_i might send an accusation of each P_j which will then be relayed by each other party. This is the flooding of at most $n^4\lambda$ bits. The output consists of at most one well-formed input of P_s so is at most $\ell + n^2\lambda$ bits. In Produce Candidate and Adopt Candidate each party sends it to at most n other parties, yielding communication at most $n^2\ell + n^4\lambda$. As for round complexity, note that if the protocol in round r runs for one more round it is because $\neg\mathsf{Complete}(S)$, which implies that $\mathsf{ToAccuse}^r(S) \neq \emptyset$ which by Item 3(a)ii in Definition 9 implies that P_k for which $(\mathsf{P}_k, \cdot) \notin \mathsf{path}$ is added to T in $\mathsf{Tree}^{\cdot}$, extending path by length 1. After this path contains r unique corrupted parties. Therefore $r \leq f$. □

The bulk for the communication of PC is the flooding of up to n^2 accusations. It turns out this can be compressed accross multiple runs of PC as each accusation need only be sent once.

Lemma 6 (Amortized Communication Complexity). *Assume that in the life time of the system ι instances $\mathsf{PC}^1, \ldots, \mathsf{PC}^\iota$ are run and that PC^i has inputs (m^i, π^i) with $|(m^i, \hat{\pi}^i)| \leq \ell_i$, where $\hat{\pi}^i$ is π^i with all accusations removed. Then the communication complexity of running all ι copies can be compressed to $\mathcal{O}\left(n^2 \sum_i \ell_i + n^4\lambda\right)$ without affecting the security of the protocol.*

Proof. If in a given system an accusation $e = (\text{Acc}, \mathsf{P}_i, \mathsf{P}_j, \sigma_i)$ was sent as part of running one PC, then do not send it again. Also, add it to all sets S_i in all copies. Also, do not send it as part of the justifications after it was sent once. If $e = (\text{Acc}, \mathsf{P}_i, \mathsf{P}_j, \sigma_i)$ was received once then add it to all incoming justifactions. This can be seen to not affect the execution of the protocol. If π justified m then it also does so after adding one more accusation. This way, overall, each of the n^2 possible $e = (\text{Acc}, \mathsf{P}_i, \mathsf{P}_j, \sigma_i)$ will be sent at most once per pair of parties for a total of $n^4\lambda$ communication. □

5 Generic Transferable Justifiers

Our second to last general tool is generic transferable justifiers. This is a short hand capturing the idea that a message can be justified by sending along all messages used to compute it and let the receiver recompute the message. In all protocols Π which follow, the protocols proceed in super rounds, where in each super round $s = 1, 2, \ldots$ the parties invoke a sub-protocol Π^s, wait for its outputs, and then run the next super round. In the first super round, we assume that each P_i has an input (x_i, π_i), where $J_{\mathsf{In}}(\mathsf{P}_i, x_i, \pi_i) = \top$ and that this is the input to the first sub-protocol Π^1. As a sentinel, let $x_i^0 = x_i$ and $\pi_i^0 = \pi_i$ and define $\Pi^0.J_{\mathsf{Out}} \coloneqq \Pi.J_{\mathsf{In}}$. Consider then a super round s where in the previous $s-1$ super rounds the parties ran protocols $\Pi^1, \ldots, \Pi^{s-1}$ and in super round s are to run Π^s. For $k = 1, \ldots, s-1$, let y_i^k be the output of P_i from Π^k and let π^k be the justifier. Then the message to be input to Π^s by P_i will be computed using a function

$$x_i^s = \mathsf{NxtInp}(\{y_i^k\}_{k=0}^{s-1}) \tag{1}$$

and the accompanying justifier computed using a function

$$\pi_i^s = \mathsf{NxtJst}(\{(y_i^k, \pi_i^k)\}_{k=0}^{s-1}) \ . \tag{2}$$

Definition 12 (Generic Justifier). *When we say that we use a generic justifier predicate in a setting as described above then we mean that*

$$\mathsf{NxtJst}(\{(y_i^k, \pi_i^k)\}_{k=0}^{s-1}) = \{(y_i^k, \pi_i^k)\}_{k=0}^{s-1} \ .$$

Furthermore, the input justifier predicate for Π^s will be

$$\Pi^s.J_{\mathsf{In}}(\mathsf{P}_j, x_i^s, \{(y_i^k, \pi_i^k)\}_{k=0}^{s-1})) \equiv \\ x_i^s = \mathsf{NxtInp}(\{y_i^k\}_{k=0}^{s-1}) \ \wedge \ \bigwedge_{k=1}^{s-1} \Pi^k.J_{\mathsf{Out}}(\mathsf{P}_j, y_i^k, \pi_i^k) \ .$$

We also use generic justifiers for the outputs. We simply show that the output can be computed from justified outputs of the sub-protocols.

Definition 13 (Generic Justifier for Output). *In a protocol with σ super-rounds, we give generic justified outputs by computing $x_i^{\sigma+1}$ as if it was an input for a virtual round $\sigma + 1$ (which we can also think of as the first round of the next protocol where $x_i^{\sigma+1}$ is input) and then we let the output of P_i be $y_i = x_i^{\sigma+1}$ and $\pi_i = \pi_i^{\sigma+1}$ where $x_i^{\sigma+1}$ is computed as in Eq. (1) and π_i is computed as in Eq. (2).*

When an output is generically justified, then from the justifier one can extract the view of the protocol execution of the party producing it. We capture this in the following definition.

Definition 14 (Unfolded View). *Consider an AJO y in some σ-super-round protocol Π as described above using generic justifiers.*[3] *Let π' be the justifier. By the* unfolded view *of y we mean*

$$\mathsf{unfold}(y, \pi') := (\,(x, \pi), (y^1, \pi^1) \ldots, (y^\sigma, \pi^\sigma), y\,) \ ,$$

where by construction π' contains $x = x^0$ and $\pi = \pi^0$ such that $\Pi.J_{\mathsf{In}}(x, \pi) = \top$ and outputs $y^1, \ldots, y^\sigma$ of its sub-protocols $\Pi^1, \ldots, \Pi^\sigma$ along with justifiers $\pi^1, \ldots, \pi^\sigma$ such that $\Pi^s.J_{\mathsf{Out}}(x^s, \pi^s) = \top$ and such that $y = \mathsf{NxtInp}(\{y^k\}_{k=0}^{\sigma})$[4] *is a correctly constructed output.*

Remark 4 (Inconsistent Unfolded Views). Note that the unfolded view does not demonstrate that the input x^s to Π^s was computed according to the protocol from $(x^0, \ldots, x^{s-1})$. It only shows that the *output* y^s which is included in the justifier π' for Π^s was justified and the the final y was computed from x and these justified y^s. In particular, if we were to unfold y^s it might give an input $x^{s\prime}$ where $x^{s\prime} \neq x^s$. This is intended and seems crucial in controlling that the size of generic justifiers does not grow exponentially. It is also a desired feature that y^s might not be consistent with $x^s = \mathsf{NxtInp}(\{y^k\}_{k=0}^{s-1})$ in the view of P_i. This will soon allow us that P_i takes over a justified output y_j^s from another party P_j for some sub-protocol, i.e., it lets $y_i^s = y_j^s$ without having to recursively check consistency of the output it takes over.

6 Send Transferable Messages with Agreement and Justified Grade Cast

We now show how to add agreement to any STM protocol by giving an STMA protocol using STM as sub-protocol. The protocol is given in Fig. 5.

Theorem 2. *Protocol $\mathsf{STMA}_{J_{\mathsf{Msg}}, \mathsf{P}_s}$ is an STMA protocol. Assume that STMA has inputs (m, π) with $|m| \leq \ell$. Then the protocol has communication complexity $\mathcal{O}(n^2\ell + n^4\lambda)$ and the size of the justified output is at most $\mathcal{O}(\ell + n^2\lambda)$. The amortized complexity can be optimised to be as in Lemma 6. If P_s is honest, it uses 2 rounds.*

Proof. All properties but validity and agreement are trivial.

Lemma 7 (Validity). *Protocol $\mathsf{STMA}_{J_{\mathsf{Msg}}, \mathsf{P}_s}$ in Fig. 5 is valid.*

Proof. Suppose P_s is honest and has input (m, π_{MSG}). In this case, P_s inputs (m, π_{MSG}) to $\mathsf{STM}_{J_{\mathsf{Msg}}, \mathsf{P}_s}$ such that $J_{\mathsf{Msg}}(P_s, m, \pi_{\mathrm{MSG}}) = \top$ and hence $J_{\mathsf{Msg}}(m, \pi_{\mathrm{MSG}}) = \top$ as P_s is honest.[5] By validity of

[3] Recall that this means that some honest party would accept y.

[4] Recall that we compute the output as if it was the input for a next virtual round. When the output is given as input to another protocol, it actually is an input for a real next round.

[5] Recall that $J(x, \pi) = \top$ denotes that $J(\mathsf{P}_i, x, \pi) = \top$ for all honest P_i, which holds if it holds for a single P_i (cf. Definition 3).

$\mathsf{STMA}_{J_{\mathsf{Msg}},\mathsf{P}_s}(m, \pi_{\mathbf{Msg}})$

The input of P_s is (m, π_{MSG}), where $J_{\mathsf{Msg}}(m, \pi_{\text{MSG}}) = \top$.

- P_s: On input (m, π_{MSG}) the sender inputs (m, π_{MSG}) to $\mathsf{STM}_{J_{\mathsf{Msg}},\mathsf{P}_s}$.
- P_i: Let y be the output of $\mathsf{STM}_{J_{\mathsf{Msg}},\mathsf{P}_s}$ and let

$$z = \begin{cases} m' & \text{if } y = (m', \pi') \text{ for } m' \neq \textsc{NoMsg} \\ \textsc{NoMsg}^{(s)} & \text{if } y = (\textsc{NoMsg}, \pi') \ . \end{cases}$$

  Here, $\textsc{NoMsg}^{(s)} \neq \textsc{NoMsg}$ is a special symbol denoting that P_s sent $\textsc{NoMsg}$ (and thus was corrupt). Let the justifier be $\gamma = y$ and let J denote the generic justifier predicate.
- P_i: Input (z, γ) to $\mathsf{STM}_{J,\mathsf{P}_i}$.
- P_i: For $j \in [n]$, let $z_j = (m_j, \gamma_j)$ be the output from $\mathsf{STM}_{J,\mathsf{P}_j}$ and let

$$\mathcal{A} = \{m_j\}_{j=1}^n \setminus \{\textsc{NoMsg}\} \ ,$$

$$m = \begin{cases} \textsc{NoMsg} & \text{if } \mathcal{A} = \{\textsc{NoMsg}^{(s)}\} \vee |\mathcal{A}| > 1 \\ m' & \text{if } \mathcal{A} = \{m'\} \text{ for } m' \neq \textsc{NoMsg}^{(s)} \ . \end{cases}$$

  Let $\delta = \{z_j\}_{\mathsf{P}_j \in \mathbb{P}}$ be the generic justifier and let $\mathsf{STMA}_{J_{\mathsf{Msg}},\mathsf{P}_s}.J_{\mathsf{Out}}$ be the generic justifier predicate.
- P_i: Output (m, δ).

Fig. 5. An STMA protocol.

$\mathsf{STM}_{J_{\mathsf{Msg}},\mathsf{P}_s}$, every AJO is of the form $y = (m, \pi')$ with $J_{\mathsf{Msg}}(m, \pi') = \top$ and all honest parties get an output. Hence, all honest parties P_i set $\gamma_i = (m, \pi'), z = m$ and input (z, γ) to $\mathsf{STM}_{J,\mathsf{P}_i}$. By validity of $\mathsf{STM}_{J,\mathsf{P}_j}$, every AJO (m, γ_j) has $J(m, \gamma_j) = \top$ when P_j is honest. For all dishonest P_j, the properties of the generic justifier and the justified messages property of $\mathsf{STM}_{J,\mathsf{P}_j}$ implies that for all AJOs z_j either $z_j = (m, \gamma_j)$ s.t. $J_{\mathsf{Msg}}(m, \gamma_j) = \top$ or $z_j = (\textsc{NoMsg}, \gamma_j)$ s.t. $\mathsf{STM}_{J,\mathsf{P}_j}.J_{\mathsf{Out}}(m, \gamma_j) = \top$. Hence, for all honest P_i, $m_j = m$ for all honest P_j and $m_j \in \{m, \textsc{NoMsg}\}$ for all dishonest P_j. Note that all AJOs must include $\delta = \{z_j\}_{\mathsf{P}_j \in \mathbb{P}}$ and be valid with respect to the generic justifier. The properties of the generic justifier therefore imply that the unfolded view of any AJO must have $\mathcal{A} = \{m\}$ and hence be of the form (m, δ). □

Lemma 8 (Agreement). *Protocol* $\mathsf{STMA}_{J_{\mathsf{Msg}},\mathsf{P}_s}$ *in Fig. 5 has agreement.*

Proof. By the properties of STM, any AJO of $\mathsf{STM}_{J_{\mathsf{Msg}},\mathsf{P}_s}$ is justified by $\mathsf{STM}_{J_{\mathsf{Msg}},\mathsf{P}_s}.J_{\mathsf{Out}}$. We first show that if there exists an AJO $(m' \neq \textsc{NoMsg}, \pi')$ from $\mathsf{STM}_{J_{\mathsf{Msg}},\mathsf{P}_s}$, then any AJO must be of the form (m', δ) or $(\textsc{NoMsg}, \delta)$, where $\delta = \{z_j\}_{P_j \in \mathbb{P}}$ s.t. the output is valid w.r.t. the generic justifier. To see this, note that by validity of $\mathsf{STM}_{J,\mathsf{P}_i}$, all its AJOs must be of the form $z_i = (m', \gamma_i)$. By the properties of the generic justifier, any party P_j must include z_i in the justifier of its output (m, δ) in order for it to be justifiable toward any honest party. Hence, the unfolded view of any AJO must form its output based on $\mathcal{A}$ that always includes m'. This shows that only (m', δ) and $(\textsc{NoMsg}, \delta)$ can be justifiable outputs for P_j in this case.

The case where there exist a justified output $(\textsc{NoMsg}, \pi)$ from $\mathsf{STM}_{J_{\mathsf{Msg}},\mathsf{P}_s}$ is similar. In this case, validity of $\mathsf{STM}_{J,\mathsf{P}_i}$ and the properties of the generic justifier together instead imply that any AJO must include $\textsc{NoMsg}^{(s)}$ in its set $\mathcal{A}$ in order for its output to be generically justifiable. Hence, all AJOs will have $m = \textsc{NoMsg}$. □

As for communication, just observe that we run STM twice on inputs of length $\mathcal{O}(\ell)$. □

6.1 Justified Grade Cast

In STMA, some parties might have output NoMsg while some have $m \neq$ NoMsg. We now show how to upgrade an STMA to a graded STM where the output contains a grade $g \in \{0, 1, 2\}$ which indicates the confidence in this output. When $g = 2$, no AJO can have $m \neq$ NoMsg. Furthermore, grades are at most 1 apart and honest senders always produce grade 2. A detailed definition is given in Definition 15. Our protocol is given in Fig. 6. It uses STMA as (blackbox) sub-protocol.

$\mathsf{GSTM}_{J_{\mathsf{Msg}},\mathsf{P}_s}(m, \pi_{\mathbf{Msg}})$

The input of P_s is (m, π_{Msg}), where $J_{\mathsf{Msg}}(m, \pi_{\text{Msg}}) = \top$.

- P_s: Input (m, π_{Msg}) to $\mathsf{STMA}_{J_{\mathsf{Msg}},\mathsf{P}_s}$.
- P_i: Let y be the output of $\mathsf{STMA}_{J_{\mathsf{Msg}},\mathsf{P}_s}$ and let

$$z = \begin{cases} m' & \text{if } y = (m', \pi') \text{ for } m' \neq \text{NoMsg} \\ \text{NoMsg}^{(s)} & \text{if } y = (\text{NoMsg}, \pi') \ . \end{cases}$$

  Let the justifier be $\gamma = y$ and let J be the generic justifier predicate.
- P_i: Input (z, γ) to $\mathsf{STMA}_{J,\mathsf{P}_i}$.
- P_i: For $j \in [n]$, let $z_j = (m_j, \gamma_j)$ be the output from $\mathsf{STMA}_{J,\mathsf{P}_j}$ and let

$$\mathcal{A} = \{m_j\}_{j=1}^{n} \setminus \{\text{NoMsg}\} \ ,$$

$$(m, g) = \begin{cases} (\text{NoMsg}, 0) & \text{if } \mathcal{A} = \{\text{NoMsg}^{(s)}\} \\ (m', 1) & \text{if } \mathcal{A} = \{m', \text{NoMsg}^{(s)}\} \text{ for } m' \neq \text{NoMsg}^{(s)} \\ (m', 2) & \text{if } \mathcal{A} = \{m'\} \text{ for } m' \neq \text{NoMsg}^{(s)} \ . \end{cases}$$

  Let $\delta = \{z_j\}_{\mathsf{P}_j \in \mathbb{P}}$ be the generic justifier and let $\mathsf{GSTM}_{J_{\mathsf{Msg}},\mathsf{P}_s}.J_{\mathsf{Out}}$ be the generic justifier predicate.
- P_i: Output $((m, g), \delta)$.

Fig. 6. A justified gradecast protocol.

Definition 15 (Graded Send Transferable Message). *Let Π be a protocol run among n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ and let J_{Msg} be a parametrisable input justifier predicate. Assume that Π specifies a designated sender P_s holding input (m, π_{Msg}) such that $J_{\mathsf{Msg}}(\mathsf{P}_s, m, \pi_{\text{Msg}}) = \top$ and parties terminate upon generating output in Π. As part of its code the protocol specifies an output justifier predicate $\Pi.J_{\mathsf{Out}}$. We say that Π is a* graded send transferable message (GSTM) protocol *if it has the following properties:*

- ***Correctness:*** *Honest P_i outputs $y_i = ((m_i, g_i), \pi_i)$ for $g_i \in \{1, 2\}$ and $m_i \neq$ NoMsg or $y_i = ((m_i = \text{NoMsg}, g_i = 0), \pi_i)$.*
- ***Justified Output:*** *When honest P_i outputs $y_i = ((m_i, g_i), \pi_i)$ then $J_{\mathsf{Out}}(\mathsf{P}_i, (m_i, g_i), \pi_i) = \top$.*
- ***Justified Message:*** *For all AJOs $y_i = ((m_i, g_i), \pi_i)$ either $m_i =$ NoMsg and $g_i = 0$ or π_i is of the form $\pi_i = (\pi_i^1, \pi_i^2)$ and $J_{\mathsf{Msg}}(m_i, \pi_i^1) = \top$.*
- ***Validity:*** *If P_s is honest and has input (m, π_{Msg}), then all honest parties P_j have output $m_j = m$ and $g_j = 2$. Furthermore, for all AJOs (m', g') it holds that $m' = m$ and $g' = 2$.*
- ***Graded Agreement:*** *For all AJOs (m^1, g^1) and (m^2, g^2) it holds that $|g^1 - g^2| \leq 1$ and if $g^1, g^2 > 0$ then $m^1 = m^2$.*

Theorem 3. *Protocol* $\mathsf{GSTM}_{J_{\mathsf{Msg}},\mathsf{P}_s}$ *in a GSTM for* $t < n$*. Assume that* GSTM *has inputs* (m, π) *with* $|m| \leq \ell$*. Then the protocol has communication complexity* $\mathcal{O}(n^2\ell + n^4\lambda)$ *and the size of the justified output is at most* $\mathcal{O}(\ell + n^2\lambda)$*. The amortized complexity can be optimised to be as in Lemma 6.*

As for STMA, all properties but validity and agreement are trivial and so is the communicaton complexity.

Lemma 9 (Validity). *Protocol* $\mathsf{GSTM}_{J_{\mathsf{Msg}},\mathsf{P}_s}$ *in Figure 6 is valid.*

Proof. Suppose that P_s is honest and has input m. By validity of $\mathsf{STMA}_{J_{\mathsf{Msg}},\mathsf{P}_s}$, every AJO y for that protocol is $y = (m', \pi')$ for $m' = m$. Now let $((m, g), \delta)$ be any AJO of $\mathsf{GSTM}_{J_{\mathsf{Msg}},\mathsf{P}_s}(m, \pi_{\text{MSG}})$ and assume for the sake of contradiction that $g \in \{0, 1\}$. Then the unfolded view of $((m, g), \delta)$ contains an AJO for $(\text{NoMsg}, 0)$ or $(m', 1)$ from $\mathsf{STMA}_{J,\mathsf{P}_j}$. By definition of $\mathcal{A}$ the unfolded view of either of these AJOs will contain an AJO for $y = (\text{NoMsg}, \pi')$ from $\mathsf{STMA}_{J_{\mathsf{Msg}},\_s}$, a contradiction. ☐

Lemma 10 (Graded Agreement). *Protocol* $\mathsf{GSTM}_{J_{\mathsf{Msg}},\mathsf{P}_s}$ *in Figure 6 has graded agreement.*

Proof. Suppose we have AJOs $((m_i, g_i), \delta_i)$ and $((m_j, g_j), \delta_j)$ with $g_i, g_j > 0$. By the properties of the generic justifier, it must be the case that the unfolded view of $((m_i, g_i), \delta_i)$ holds an AJO for $m_i \neq \text{NoMsg}$ from some $\mathsf{STMA}_{J,\mathsf{P}_k}$. By the justified message property of STMA it follows that m_i is justified using the input justifier J of $\mathsf{STMA}_{J,\mathsf{P}_k}$, which was the genetic justifier checking that m_i was an AJO of $\mathsf{STMA}_{J_{\mathsf{Msg}},\mathsf{P}_s}$. Symmetrically, we can conclude that m_j was an AJO of $\mathsf{STMA}_{J_{\mathsf{Msg}},\mathsf{P}_s}$. By the agreement property of $\mathsf{STMA}_{J_{\mathsf{Msg}},\mathsf{P}_s}$ it follows that $m_i = m_j$. It remains to show that it cannot be the case for two AJOs that $g_i = 2$ and $g_j = 0$. Assume that $g_i = 0$. Suppose then for the sake of contradiction that we have AJOs $((m_i, g_i), \delta_i)$ and $((m_j, g_j), \delta_j)$ with $g_i = 0$ and $g_j = 2$. The unfolded view of $((m_i, g_i), \delta_i)$ holds an AJO for $m_i = \text{NoMsg}^{(s)}$ from all $\mathsf{STMA}_{J,\mathsf{P}_k}$ which did not output NoMsg, in particular for all honest P_k. Symmetrically, we can conclude that m_j was an AJO from $\mathsf{STMA}_{J,\mathsf{P}_k}$ for all honest P_j. Since $g_j = 2$ we have $m_j \neq \text{NoMsg}^{(s)}$. Since there is at least one honest P_j we have found one $\mathsf{STMA}_{J,\mathsf{P}_k}$ where P_k is honest and it has AJO $\text{NoMsg}^{(s)}$ and AJO $m_j \neq \text{NoMsg}^{(s)}$. This breaks validity of $\mathsf{STMA}_{J,\mathsf{P}_k}$ as the honest P_k cannot have inputted both $\text{NoMsg}^{(s)}$ and $\neq \text{NoMsg}^{(s)}$. ☐

7 Early Stopping Broadcast

We now present our main result, early stopping broadcast.

7.1 Diagonal Cast

We begin by building an early stopping protocol, called DC (for *diagonal cast*), which is early stopping, but may run up to $O(t^2)$ rounds. The protocol will be a justifiable broadcast in the sense of Definition 7, which implies that it is also a broadcast protoccol. The protocol DC uses GSTM as (blackbox) sub-protocol. Since we use DC as a building block in our final protocol, we also make its output justified. During the protocol we use a helper function computing a party's next vote. In each round the parties run one GSTM to get output (m, g). Consider a party which in the previous rounds saw outputs $(m^1, g^1), \ldots, (m^r, g^r)$. Then it should use the m from the latest GSTM with

$g^\rho > 0$. If no such ρ exists it should use $\textsc{NoMsg}^{(s)}$ to indicate the sender P_s was corrupt. More formally we use this function:

$$\mathsf{NxtInp}((m^1, g^1), \ldots, (m^r, g^r)) = \begin{cases} \textsc{NoMsg}^{(s)} & \text{if } g^1 = \cdots = g^r = 0 \\ m_{\max\{i \in [r] \,|\, g_i > 0\}} & \text{otherwise.} \end{cases}$$

Below we let $i^* = \max\{i \in [r] \,|\, g_i > 0\}$ when we are not in the case $g^1 = \cdots = g^r = 0$. At the end of the protocol we want to map $\textsc{NoMsg}^{(s)}$ to $\textsc{NoMsg}$. For this we use a simple helper: $\mathsf{Out}(m) = \textsc{NoMsg}$ if $m = \textsc{NoMsg}^{(s)}$ and $\mathsf{Out}(m) = m$ otherwise. The protocol is given in Fig. 7.

Diagonal Cast $\mathsf{DC}_{J_{\mathsf{Msg}}, \mathsf{P}_s}$

Without loss of generality, assume that P_1 is the sender, i.e., $s = 1$.

- In round 1:
  - P_1: Has input $(m, \pi_{\textsc{Msg}})$ with $J_{\mathsf{Msg}}(\mathsf{P}_1, m, \pi_{\textsc{Msg}}) = \top$ and inputs it to $\mathsf{GSTM}_{J_{\mathsf{Msg}}, \mathsf{P}_1}$.
  - P_i: Let y be the output of P_i and parse it as $y = ((m_i^1, g_i^1), \pi_i^1)$.
  - P_i: If $g_i^1 = 2$ then send $(1, (m_i^1, g_i^1), \pi_i^1)$ to all other parties, output $m = \mathsf{Out}(m_i^1)$, and run for one more round and then terminate.[a]
- For $j \in \{2, \ldots, n\}$ do as follows:
  - P_j: Compute $m_j = \mathsf{NxtInp}((m_j^1, g_j^1), \ldots, (m_j^{j-1}, g_j^{j-1}))$.
  - P_j: Compute the generic justifier $\gamma_j = \{((m_j^k, g_j^k), \pi_j^k)\}_{k=1}^{j-1}$ and let J^j be the generic justifier predicate for this round.
  - P_j: Input (m_j, γ_j) to $\mathsf{GSTM}_{J^j, \mathsf{P}_j}$.
  - P_i: Let y be the output of party $\mathsf{GSTM}_{J^j, \mathsf{P}_j}$ and parse it as $y = ((m_i^j, g_i^j), \pi_i^j)$.
  - P_i: If $g_i^j = 2$ then send $(j, (m_i^j, g_i^j), \pi_i^j)$ to all other parties, output $m = \mathsf{Out}(m_i^j)$, and run for one more round and then terminate.
- P_i: Upon receiving a tuple $(j, (m', g'), \pi')$ (for any $j \in [n]$ and in any round) such that $\mathsf{GSTM}_{J^j, \mathsf{P}_j}.J_{\mathsf{Out}}(\mathsf{P}_i, (m', g'), \pi') = \top$ and $g' = 2$, forward it to all parties, output $m = \mathsf{Out}(m')$, and run for one more round and then terminate.
- The output justifier for the protocol is $\mathsf{DC}_{J_{\mathsf{Msg}}, \mathsf{P}_s}.J_{\mathsf{Out}}(m, \pi)$, which is true if $\pi = (j, (m', g'), \pi')$ and $j \in [n]$ and $\mathsf{GSTM}_{J^j, \mathsf{P}_j}.J_{\mathsf{Out}}(\mathsf{P}_i, (m', g'), \pi') = \top$ and $m = \mathsf{Out}(m')$.

[a] To ensure that no honest party terminates until all honest gave output.

Fig. 7. The broadcast protocol.

Lemma 11 (Validity). DC *satisfies validity.*

Proof. Suppose that P_1 is an honest sender holding input $(m, \pi_{\textsc{Msg}})$. Since $J_{\mathsf{Msg}}(\mathsf{P}_1, m, \pi_{\textsc{Msg}}) = \top$ validity of $\mathsf{GSTM}_{J_{\mathsf{Msg}}, \mathsf{P}_1}$ implies that all honest parties get output $(m, 2, \pi)$. Hence, all parties terminate with output m after running for one more iteration of the loop. □

Lemma 12 (Stabilisation). *Let j be the minimal iteration such that for some honest P_i an AJO $((m, 2), \pi)$ satisfying $\mathsf{GSTM}_{J^j, \mathsf{P}_j}.J_{\mathsf{Out}}(\mathsf{P}_i, (m, 2), \pi) = \top$ can be produced. Then for any $j' \geq j$ and any honest $\mathsf{P}_{i'}$ and any AJO $((m', 2), \pi')$ satisfying $\mathsf{GSTM}_{J^{j'}, \mathsf{P}_{j'}}.J_{\mathsf{Out}}(\mathsf{P}_{i'}, (m', g'), \pi') = \top$ it holds that $g' = 0$ or $m' = m$. Furthermore, if $j' = j$ then $g' > 0$ and thus $m' = m$.*

Proof. We do induction in $|j' - j| = 0, 1, \ldots$. For the base case $|j' - j| = 0$ we have $j' = j$ and that $g' \in \{1, 2\}$ and $m' = m$ by graded agreement of $\mathsf{GSTM}_{J^j, \mathsf{P}_j}$. Assume then the induction hypothesis for all $|j' - j| < \ell$. We prove it for $|j' - j| = \ell$. We have to prove that $g' = 0$ or $m = m'$. So it is enough to prove that if $g' > 0$ then $m' = m$. When $g' > 0$ then by the justified message property of $\mathsf{GSTM}_{J^{j'},\mathsf{P}_{j'}}$ we have that m' is justified by $J^{j'}$, which was the generic justifier for iteration j'. Therefore m' was computed as

$$m' = \mathsf{NxtInp}((m^1, g^1), \ldots, (m^{j'-1}, g^{j'-1}))$$

from AJOs. By induction hypothesis $g^j \geq 1$ and $m^j = m$, and for $j \leq k \leq j'-1$ it holds that $g^k = 0$ or $m^k = m$. This by construction gives $\mathsf{NxtInp}((m^1, g^1), \ldots, (m^{j'-1}, g^{j'-1})) = m$, as desired. □

Corollary 2. DC *has agreement.*

Proof. When an honest party P_i produces output m' then it by construction produces or receives an AJO $(m', g = 2)$ for some $\mathsf{GSTM}_{J^{j'},\mathsf{P}_{j'}}$. Clearly there is a smallest j for which an AJO $(m, g = 2)$ can be produced for $\mathsf{GSTM}_{J^j,\mathsf{P}_j}$. By Lemma 12 it holds that $m' = m$. This holds for all honest outputs m'. □

Lemma 13. DC *terminates in* $8(f + 1)^2$ *rounds. If* P_1 *is honest then it runs in at most* $8(f + 1)$ *rounds.*

Proof. The protocol terminates at the latest once the first honest party acts as the sender in some iteration j as this gives $g_j = 2$. This is worstcase in iteration $(f + 1)$. Each iteration runs one GSTM, which uses two STMA, which each uses two STM, which each uses $f + 1$ rounds. This gives $4(f + 1)$ rounds per GSTM. Applying the staggering compiler then blows up the round complexity by a factor of 2. □

7.2 Weak Early Stopping $\mathcal{O}(f)$ and Worstcase $\mathcal{O}(t)$

The protocol DC can do early stopping, but if there are $f = \omega(\sqrt{t})$ corruptions it runs for more than $\mathcal{O}(t)$ rounds, which is asymptotically sub-optimal. We solve this by capping the running time at $\mathcal{O}(t)$. Doing this safely is subtle, and we now present a protocol with weak early stopping which helps doing it safely. Weak early stopping means that the protocol achieves early stopping when the sender is honest. If the sender is corrupt it may run for $\mathcal{O}(n)$ rounds. We describe the protocol with P_1 as sender, but it can be adopted to any P_s.

$\mathsf{WES}_{J_{\mathsf{Msg}},\mathsf{P}_1}$

- P_1: Has input (m, π_{MSG}) with $J_{\mathsf{Msg}}(\mathsf{P}_1, m, \pi_{\text{MSG}}) = \top$ and inputs it to $\mathsf{GSTM}_{J_{\mathsf{Msg}},\mathsf{P}_1}$.
- P_1: Let $((m, g), \pi)$ be the output of $\mathsf{GSTM}_{J_{\mathsf{Msg}},\mathsf{P}_1}$ and run $\mathsf{DSC}_{\mathsf{P}_1,J}$ with input $((m, g), \pi)$ by P_1, where J is the generic justifier predicate.
- P_i: Upon receiving a tuple $((m', g'), \pi')$ as output from $\mathsf{GSTM}_{J_{\mathsf{Msg}},\mathsf{P}_1}$ or relayed by another party, where $\mathsf{GSTM}_{J_{\mathsf{Msg}},\mathsf{P}_1}.J_{\mathsf{Out}}(\mathsf{P}_i, (m', g'), \pi') = \top$ and $g' = 2$, forward it to all parties, output $m = m'$ and terminate.
- If $\mathsf{DSC}_{\mathsf{P}_1,J_{\mathsf{Msg}}}$ produces output $((m, g), \pi)$ before the output was produced using the above rule, then output m $\mathsf{GSTM}_{J_{\mathsf{Msg}},\mathsf{P}_1}.J_{\mathsf{Out}}(\mathsf{P}_i, (m, g), \pi) = \top$ and $g > 0$ and NoMsg otherwise.

Fig. 8. The weak early stopping broadcast protocol.

Theorem 4. *The protocol* $\mathsf{WES}_{J_{\mathsf{Msg}},\mathsf{P}_1}$ *is a broadcast protocol. Parties terminate one round apart and the round complexity is* $\mathcal{O}(t)$*. If* P_1 *is honest then the round complexity is* $\mathcal{O}(f)$*.*

Proof. Validity is trivial: if P_1 is honest then $\mathsf{GSTM}_{J_{\mathsf{Msg}},\mathsf{P}_1}$ outputs $((m,g),\pi)$ with $g=2$ and all honest parties output m. This happens within one run of GSTM, so in $\mathcal{O}(f)$ rounds. We argue agreement. If all honest parties give output by receiving a tuple $((m',g'),\pi')$ which is an AJO for $\mathsf{GSTM}_{J_{\mathsf{Msg}},\mathsf{P}_1}$ and with $g'=2$ then agreement follows from agreement of $\mathsf{GSTM}_{J_{\mathsf{Msg}},\mathsf{P}_1}$. If all honest parties give output by receiving $((m,g),\pi)$ from DSC then agreement follows from agreement of DSC. Assume then that some honest P_i gives output using $((m'=m_i,g'=2),\pi')$ which is an AJO for $\mathsf{GSTM}_{J_{\mathsf{Msg}},\mathsf{P}_1}$ and some honest P_j gives output m_j by receiving $((m,g),\pi)$ from $\mathsf{DSC}_{\mathsf{P}_1,J}$. Since $g'=2$ and $m'=m_i$ it follows from graded agreement for $\mathsf{GSTM}_{J_{\mathsf{Msg}},\mathsf{P}_1}$ that $m=m_i$ and $g>0$ for all AJOs for $\mathsf{GSTM}_{J_{\mathsf{Msg}},\mathsf{P}_1}$. Since J is the generic justifier it follows that all AJOs $((m,g),\pi)$ for $\mathsf{DSC}_{\mathsf{P}_1,J}$ has $m''=m_i$ and $g''>0$. Therefore P_j has output $m_j=m=m_i$. Parties terminate one round apart as they terminate one round apart in DSC and if they terminate by the $g'=2$ rule then they forward the AJO and then all honest parties terminate in the next round. □

7.3 Early Stopping $\mathcal{O}(f^2)$ and Worstcase $\mathcal{O}(t)$

We now give a broadcast protocol Capped Diagonal Cast which basically runs Diagonal Cast for a capped number of rounds and uses WES to have all parties report whether they saw an output from DC before the time cap. Since the reports are sent with WES parties will agree on the reports and make the same decision. Furthermore, if an honest party saw an output it will be reported with early stopping. We again describe the protocol with P_1 as sender, but can trivially adopt to any P_s.

Capped Diagonal Cast $\mathsf{CDC}_{\mathsf{P}_1,J_{\mathsf{Msg}}}$

We describe the protocol from the view of party P_i.

- P_1: Has input (m,π_{MSG}) with $J_{\mathsf{Msg}}(m,\pi_{\text{MSG}})=\top$ and inputs it to $\mathsf{DC}_{\mathsf{P}_1,J_{\mathsf{Msg}}}$.
- P_i: Participate in $\mathsf{DC}_{\mathsf{P}_1,J_{\mathsf{Msg}}}$ for at most $8(t+1)$ rounds.[a]
- P_i: If $\mathsf{DC}_{\mathsf{P}_1,J_{\mathsf{Msg}}}$ produced output (m,π) in or before round $8n$ then run $\mathsf{WES}_{\mathsf{P}_i,\top}$ with input (m,π) in the round where $\mathsf{DC}_{\mathsf{P}_1,J_{\mathsf{Msg}}}$ produced output. If by round $8(t+1)$ protocol $\mathsf{DC}_{\mathsf{P}_1,J_{\mathsf{Msg}}}$ did not produce output run $\mathsf{WES}_{\mathsf{P}_i,\top}$ with input NoMsg in the round $8(t+1)$.
- P_i: If and when the first $\mathsf{WES}_{\mathsf{P}_j,\top}$ outputs (m,π) such that $\mathsf{DC}_{\mathsf{P}_1,J_{\mathsf{Msg}}}.J_{\mathsf{Out}}(m,\pi)=\top$, output m.
- P_i: If $\mathsf{WES}_{\mathsf{P}_1,\top},\dots,\mathsf{WES}_{\mathsf{P}_n,\top}$ all terminated and none had an output (m,π) such that $\mathsf{DC}_{\mathsf{P}_1,J_{\mathsf{Msg}}}.J_{\mathsf{Out}}(m,\pi)=\top$, then output NoMsg.

[a] Here we count base communication rounds, not iterations of GSTM.

Fig. 9. An Early Stopping Broadcast protocol with $\mathcal{O}(\min((f+1)^2,n))$ rounds.

Theorem 5. *The protocol* $\mathsf{CDC}_{J_{\mathsf{Msg}},\mathsf{P}_1}$ *is a broadcast protocol. Parties terminate one round apart and the round complexity is* $\mathcal{O}(\min(f^2,t))$*. If the sender is honest output is given in* $\mathcal{O}(f)$ *rounds. From* $\mathsf{CDC}_{J_{\mathsf{Msg}},\mathsf{P}_1}$ *we can get a broadcast protocol* EBC *for definition Definition 1 with the same communication complexity simply by dropping the input justifier predicate* J_{Msg} *and the justifier* π_{MSG}*.*

Proof. First note that all $\mathsf{WES}_{\mathsf{P}_i,\top}$ are started at most one round apart. Namely, no party starts them later then by round $8(t+1)$. So if they are to be started 2 rounds apart the first is started in round $8(t+1)-2$ or earlier. But then $\mathsf{DC}_{\mathsf{P}_1,J_{\mathsf{Msg}}}$ terminated by round $8(t+1)-2$ at the first honest party. But then it terminated by round $8(t+1)-1$ at all honest. Hence all honest started $\mathsf{WES}_{\mathsf{P}_i,\top}$ exactly when $\mathsf{DC}_{\mathsf{P}_1,J_{\mathsf{Msg}}}$ terminated, which is at most one round apart. We can then apply the staggering compiler to ensure that $\mathsf{WES}_{\mathsf{P}_i,\top}$ tolerates being started one round apart. This gives a $\mathcal{O}(1)$ blowup in round comlexity. We then argue validity: if P_1 is honest then $\mathsf{DC}_{J_{\mathsf{Msg}},\mathsf{P}_1}$ outputs (m,π) within $8(f+1) \leq 8(t+1)$ rounds. Therefore an honest party P_j will send (m,π) on $\mathsf{WES}_{\mathsf{P}_j,\top}$ which will output (m,π) within $\mathcal{O}(f)$ rounds and then all honest output m. This all happens within $\mathcal{O}(f)$ rounds. Agreement is trivial from agreement of WES, as the output is computed deterministically from the outputs of $\mathsf{WES}_{\mathsf{P}_1,\top},\ldots,\mathsf{WES}_{\mathsf{P}_n,\top}$. Consider then the round complexity. We have that all copies of WES are started after $\mathcal{O}(\min(f^2,n))$ rounds as $\mathsf{DC}_{\mathsf{P}_1,J_{\mathsf{Msg}}}$ stops after $\mathcal{O}(f^2)$ rounds and we cap after $8(t+1)$ rounds. If $\mathsf{DC}_{\mathsf{P}_1,J_{\mathsf{Msg}}}$ did give an output at all honest parties before round $8(t+1)$ then it will be input to $\mathsf{WES}_{\mathsf{P}_i,\top}$ which will output after $\mathcal{O}(f)$ rounds, and hence output is produces in $\mathcal{O}(f^2)$ rounds as $\mathsf{DC}_{\mathsf{P}_1,J_{\mathsf{Msg}}}$ terminates in $\mathcal{O}(f^2)$ rounds. If $\mathsf{DC}_{\mathsf{P}_1,J_{\mathsf{Msg}}}$ did not give output within $8(t+1)$ rounds then $8(t+1) = \mathcal{O}(f^2)$, so $\mathcal{O}(\min(f^2,t)) = \mathcal{O}(t)$. And in this case the overall protocols runs for at most $\mathcal{O}(t)$ rounds as each WES terminates in $\mathcal{O}(t)$ rounds. ☐

References

1. Ittai Abraham, T.-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In Peter Robinson and Faith Ellen, editors, *38th ACM PODC*, pages 317–326. ACM, July / August 2019.
2. Ittai Abraham and Danny Dolev. Byzantine agreement with optimal early stopping, optimal resilience and polynomial complexity. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 605–614. ACM Press, June 2015.
3. Andreea B. Alexandru, Julian Loss, Charalampos Papamanthou, and Giorgos Tsimos. Sublinear-round broadcast without trusted setup against dishonest majority. Cryptology ePrint Archive, Report 2022/1383, 2022. https://eprint.iacr.org/2022/1383.
4. Piotr Berman and Juan A. Garay. $n/4$-resilient distributed consensus in $t+1$ rounds. *Mathematical Systems Theory*, 26(1):3–19, 1993.
5. Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Towards optimal distributed consensus (extended abstract). In *30th FOCS*, pages 410–415. IEEE Computer Society Press, October / November 1989.
6. Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Optimal early stopping in distributed consensus (extended abstract). In *WDAG*, pages 221–237, 1992.
7. T.-H. Hubert Chan, Rafael Pass, and Elaine Shi. Sublinear-round byzantine agreement under corrupt majority. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 246–265. Springer, Heidelberg, May 2020.
8. Tushar Deepak Chandra and Sam Toueg. Time and message efficient reliable broadcasts. In *WDAG*, pages 289–303, 1990.
9. Danny Dolev and Christoph Lenzen. Early-deciding consensus is expensive. In Panagiota Fatourou and Gadi Taubenfeld, editors, *32nd ACM PODC*, pages 270–279. ACM, July 2013.
10. Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. In Robert L. Probert, Michael J. Fischer, and Nicola Santoro, editors, *1st ACM PODC*, pages 132–140. ACM, August 1982.
11. Danny Dolev, Rüdiger Reischuk, and H. Raymond Strong. Early stopping in byzantine agreement. *J. ACM*, 37(4):720–741, 1990.
12. Danny Dolev and H. Raymond Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.
13. Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

14. Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *20th ACM STOC*, pages 148–161. ACM Press, May 1988.
15. Matthias Fitzi and Jesper Buus Nielsen. On the number of synchronous rounds sufficient for authenticated byzantine agreement. In *DISC*, pages 449–463, 2009.
16. Juan A. Garay, Jonathan Katz, Chiu-Yuen Koo, and Rafail Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *48th FOCS*, pages 658–668. IEEE Computer Society Press, October 2007.
17. Juan A. Garay and Yoram Moses. Fully polynomial byzantine agreement in t+1 rounds. In *25th ACM STOC*, pages 31–41. ACM Press, May 1993.
18. Juan A. Garay and Yoram Moses. Fully polynomial byzantine agreement for n ¿ 3t processors in t + 1 rounds. *SIAM Journal on Computing*, 27(1):247–290, 1998.
19. Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 445–462. Springer, Heidelberg, August 2006.
20. Idit Keidar and Sergio Rajsbaum. A simple proof of the uniform consensus synchronous lower bound. *Information Processing Letters*, 85(1):47–52, 2003.
21. Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Trans. Programming Languages and Systems*, 4(3):382–401, 1982.
22. Christoph Lenzen and Sahar Sheikholeslami. A recursive early-stopping phase king protocol. In *ACM PODC*, pages 60–69. ACM, 2022.
23. Yehuda Lindell, Anna Lysyanskaya, and Tal Rabin. Sequential composition of protocols without simultaneous termination. In Aleta Ricciardi, editor, *21st ACM PODC*, pages 203–212. ACM, July 2002.
24. Silvio Micali. Very simple and efficient byzantine agreement. In Christos H. Papadimitriou, editor, *ITCS 2017*, volume 4266, pages 6:1–6:1, 67, January 2017. LIPIcs.
25. Jesper Buus Nielsen. *On Protocol Security in the Cryptographic Model.* PhD thesis, Aarhus University, 2003.
26. Philippe Raipin Parvédy and Michel Raynal. Optimal early stopping uniform consensus in synchronous systems with process omission failures. In *SPAA*, volume 302–310, 2004.
27. Kenneth J. Perry and Sam Toueg. An authenticated byzantine generals algoorithm with early stopping. Technical report, Cornell University, 1984.
28. Shravan Srinivasan, Julian Loss, Giulio Malavolta, Kartik Nayak, Charalampos Papamanthou, and Sri Aravinda Krishnan Thyagarajan. Transparent batchable time-lock puzzles and applications to byzantine consensus. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 554–584. Springer, Heidelberg, May 2023.
29. Jun Wan, Hanshen Xiao, Srinivas Devadas, and Elaine Shi. Round-efficient byzantine broadcast under strongly adaptive and majority corruptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 412–456. Springer, Heidelberg, November 2020.
30. Jun Wan, Hanshen Xiao, Elaine Shi, and Srinivas Devadas. Expected constant round byzantine broadcast under dishonest majority. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 381–411. Springer, Heidelberg, November 2020.