

On the Round Complexity of Early Stopping

(Or: How to Count Synchronous Rounds)

Wonseok Choi¹, Wookjin Choi¹, Xiangyu Liu², and Vassilis Zikas³

¹ DGIST, Korea

² CISA Helmholtz Center for Information Security, Germany

³ Georgia Institute of Technology, GA, USA

wonseok@dgist.ac.kr, wookjin.choi@dgist.ac.kr, xiangyu.liu@cispa.de, vzikas@gatech.edu

Abstract. A common pipeline of cryptographic research is: a theory paper asks an interesting question which triggers a line of works; then the problem remains dormant until technology catches up and makes the problem potentially relevant for practice; this triggers a renewed interest which at times pushes the originally theoretical ideas to practice. Several highly influential ideas have followed this pipeline: e.g., the original work on Oblivious RAMs by Goldreich and Ostrovsky [STOC’87&90], polynomial commitments by Kate, Zaverucha, and Goldberg [ASIACRYPT’10], and many others. A problem that shows early signs of this pattern is early-stopping Byzantine Agreement (esBA): esBA protocols whose round complexity depends on the number f of *actually* faulty parties rather than on (an upper bound on) the number t of *potentially* faulty parties. Although it is fair to say that esBA protocols are not (yet) within the practical realm, their potential to improve large scale distributed systems like blockchain ledgers has fueled a number of recent novel results pushing the envelope in esBA feasibility (e.g., Loss and Nielsen [EUROCRYPT’24] and Elsheimy, Loss, and Papamanthou [ASIACRYPT’24]).

The original starting point of this work has been to advance on the above feasibility questions. But as is common for problems following this trajectory, renewed interest often comes with observations about issues in the assumed definitions and models. Interestingly, our investigation revealed an issue with how the existing literature has been computing the round complexity of such protocols, which yields (previously unobserved) inconsistencies in the literature of esBA and more generally of synchronous protocols. In a nutshell, the inconsistencies stem from the fact that the last round of several esBA protocols has parties terminating as soon as they send a message to other parties, i.e., they don’t need to wait to receive messages sent to them in that round. So, should we count this as an extra round or not? As we observe, different works give different answers to this question, and the answer one adopts affects the claimed tightness of the corresponding bounds.

Building on the above observation, our work proposes a fine-grained manner of counting synchronous rounds that is both intuitive and resolves these inconsistencies. In a nutshell, our counting distinguishes between rounds in which parties might both send and receive messages, and rounds in which parties only need to send messages. Examining the existing literature through the lens of this round-counting methodology exposes gaps and new open questions in the relevant literature. We resolve these gaps and extend the landscape by several new results.

1 Introduction

The synchronous model of execution is the simplest and most theoretically appealing model for distributed protocols and cryptographic multi-party protocols. In this model protocols advance in synchronous rounds, where in each round parties can receive message and act upon message they have seen—e.g., perform some computation or send messages to other parties. The two most common theoretical instantiations posture that (1) messages sent in some round are delivered by the beginning of the next round, and (2) messages sent in some round are delivered by the end of that round. These two instantiations are typically considered equivalent, as feasibility and asymptotic efficiency questions do not depend on which of the two instantiations we chose. However, this is not the case for concrete (non-asymptotic) round-complexity results.

When counting the concrete round complexity of distributed protocols, choosing between the above two definitions can already create a discrepancy in the calculated round complexity. To see why, consider the

well-known Dolev-Strong Broadcast protocol [DS83], which allows a sender to broadcast his input message x to a set of n parties, so that every party terminates with an output y satisfying the following properties: y is the same for all honest (aka uncorrupted or non-Byzantine) parties (consistency), and it equals x (the sender’s input) if the sender is honest (validity).⁴ The Dolev-Strong Broadcast protocol tolerates an arbitrary $t < n$ number of corrupted parties; at a high level, it proceeds in rounds where in each round, every party p_i checks its received messages, and if some message has enough signatures from (a subset of) the n parties to indicate that it should be included in p_i ’s output, then p_i includes it, computes a signature to it and forwards it to the whole party set. The exact details of the predicate that leads to adding the message to one’s set is not of relevance here; what is important is that in order for the protocol to be secure, the parties need to compute their output based on messages sent in $t + 1$ (consecutive) rounds. In fact, as proven already in [DS83], when considering deterministic and simultaneous termination protocols, the above is a tight bound on the round complexity.

The discrepancies introduced by the above two instantiations of the synchronous models are apparent if one attempts to count the number of rounds for Dolev-Strong: Under the first instantiation, parties need to wait for round $t + 2$ to start (so that they receive all messages sent to them in round $t + 1$) before they finalize their output and terminate; whereas under the second instantiation, parties receive all the messages they need by the end of round $t + 1$ and can immediately terminate. So, should Dolev-Strong be considered a $(t + 1)$ -round or a $(t + 2)$ -round protocol?

The above already foggy landscape becomes even more obscure, when considering so-called *early-stopping* (synchronous) protocols. In a nutshell, such protocols, aim to relax the connection between round complexity and (the upper bound on) the number t of corruptions, by making the termination round a function of the actual number f of faulty parties. (Note that $f \leq t$ since the adversary might choose to corrupt less than t parties, or corrupt some parties but still have them run the protocol, i.e., remain non-faulty.) We note in passing, that such improved round complexity comes at the cost of sacrificing *simultaneous termination*, i.e., different (honest) parties might terminate at different rounds; therefore, in this context, round complexity counts the number of rounds that the slowest (last) honest party needs to terminate.

The original starting point of this work was to devise round-optimal early-stopping Byzantine Agreement protocols for various settings (and several of our technical results are attacking this problem). This endeavor is fueled by an increasingly popular view in distributed systems and blockchain communities that worst-case corruption models might result in too slow and less scalable protocols in real-world applications. One way to mitigate such slowdowns is to resort to more restricted adversaries (e.g., only protect against fail-crashes [FL82,DM86,PR04,PT86] or base security on assumptions about the adversary’s rationality [GKTZ12]), but discounting on the security end in this way naturally opens up the system to attacks by stronger adversaries. An alternative approach is to resort to optimistic security with worst-case fallback. The idea here is for the system to operate in a mode which is highly performant in favorable conditions (e.g., fast network conditions, perfect synchrony, weaker attackers, etc.) but remains secure (albeit less performant) in worst-case environments. Such optimistic approaches has long been the subject of study in the cryptographic community [IKK⁺11] and has been recently followed by several real-time distributed systems and proposals [PS18,BHMN21,CLOZ25]. Early-stopping protocols can be viewed as another instantiation of this idea: the protocol is secure against t corruptions, but when the number f of actual corrupted parties that deviate (i.e., faults) is much smaller than t , then the protocol is faster. This is likely the reason that despite their investigation being mostly dormant since the 90’s, we have recently witnessed a renewed interest in both cryptography and distributed computing.

As is natural with problems that stay dormant for long and become popular again due to the demand for solving them, the early-stopping landscape is fragmented and has several open problems. Interestingly, however, in the process of solving such open problems, we realized that there is a much deeper issue with the way the synchronous-protocols literature—even beyond early stopping—has counted rounds that is reflected, as we will discuss, in inconsistent statements in results published at top venues: In particular, as we shall see

⁴ For simplicity, we consider in this introduction a static adversary—where the set of corrupted parties is chosen at the beginning of the protocol—as this is already sufficient for demonstrating the issues with existing definitions of rounds.

there is an example of a protocol claimed to be round-optimal, whereas another one, with the same round complexity claimed to be suboptimal (and posting an optimal construction as an open problem). Intriguingly, due to the ambiguity in round counting, none of these claims is falsifiable.

The above situation has forced us to take a detour from our original technical goal: We first propose a different way for counting rounds that resolves the inconsistencies in the literature and allows for drawing a consistent landscape of round complexity for early-stopping Byzantine Agreement (esBA). Armed with this new notion, we re-evaluate existing literature. Our notion allows us to identify gaps in this literature that were overlooked due to the ambiguity of the previous round definition(s). This elevated the re-evaluation to an additional result of its own accord, with several technical contributions, besides the new feasibility results that we prove here. We view this as evidence both for the need for our more fine-grained definition, and as an indication that such a definition can be of independent interest for the concrete-round complexity literature, beyond just early stopping. In the following, we provide an overview of our contributions.

1.1 Overview of Our Contributions

In the following we outline the contributions of our work. We keep the discussion in this section at a relatively high level, and postpone the exposition of the technical details for our results to Section 2. We include Table 1 that summarizes the landscape of concrete (non-asymptotic) round complexity of esBA and puts our new results in perspective.

1.1.1 Inconsistencies Resulting from Existing Round-Counting Methods. The starting point of our work is to revisit constructions and bounds in the classical distributed computing literature that yield contradictory statements. We note in passing that in this work we use BA to refer to both (Byzantine) Broadcast (as discussed above) and Consensus, i.e., where instead of a sender only, each of the n parties has an input and their goal is to agree on the output y (consistency), such that if all (honest) parties have input x then $y = x$ (validity).

A classic example of early-stopping BA is the seminal work by Berman and Garay [BG93], which provided a polynomial-message Consensus protocol tolerating up to $t < n/4$ corruptions. The number of rounds claimed this is (at most) $\min\{f + 2, t + 1\}$. Similarly to Dolev-Strong, the (slowest) parties in [BG93] need to receive messages sent to them in the $\min\{f + 2, t + 1\}$ -st round to finalize their output. This again creates an ambiguity depending on whether [BG93] should be considered a $\min\{f + 2, t + 1\}$ -round or an $(\min\{f + 2, t + 1\} + 1)$ -round protocol, depending on which of the two definitions of synchronous execution we chose. However, zooming into the protocol from [BG93], one can observe that there are executions where the slowest party, say p , can compute his output based on the messages sent to it in the first $\min\{f + 2, t + 1\}$ rounds, but needs to send one more message (a special “done” signal) once he decides to terminate unless it is already in round $t + 1$. This “done” message will in-fact not be accounted for by any other party (since this party is the slowest, hence all other parties can terminate before receiving it); but because p does not know that he is the slowest one, he has to send that message. (Indeed, eliminating the message altogether renders the protocol insecure.)

So now the question becomes: given that everyone (including the slowest party p) will ignore the message sent by p after round $f + 2$, shall we consider the above a $\min\{f + 2, t + 1\}$ -round or a $\min\{f + 3, t + 1\}$ -round protocol?

By strictly interpreting the description of a round as discussed above, this should be considered as $\min\{f + 3, t + 1\}$, no matter which of the two theoretical instantiations we adopt. But one can argue that this is inaccurate, since everyone by p terminates in round $\min\{f + 2, t + 1\}$ and p himself only needs to send the message and terminate—i.e., does not need to wait for the round timeout. So, strictly speaking, if a round lasts τ seconds, considering the above a $\min\{f + 3, t + 1\}$ -round protocol is badly overestimating the protocol’s duration. In fact, this latter line of thought is likely the reason why [BG93] claimed the above protocol as a $\min\{f + 2, t + 1\}$ -round protocol. But the above ambiguity transformed into inconsistencies in follow-up works, which appear to count such “ignored” send-operation in its round complexity. For example, [GM98] has a similar terminating-round structure, where the slowest party can compute its output based on the

messages sent to it in the first $\min\{f + 3, t + 1\}$ rounds, but before termination it needs to run two more rounds to send two more messages; the claimed round complexity in [GM98] is $\min\{f + 5, t + 1\}$ instead of $\min\{f + 4, t + 1\}$, which would be inconsistent with the round counting in [BG93]. In a more recent result, [ELP25] devised a $\min\{f + 3, t + 1\}$ protocol with the above structure, i.e., the last $\min\{f + 3, t + 1\}$ -st round being an “ignored” send-round by the slowest party. Despite achieving the same round-complexity as [BG93], which is claimed to be optimal [BG93], this latter result flags itself as suboptimal and posts an optimal construction as an open problem.

To summarize, [BG93] and [ELP25] have the same round complexity under the same counting convention as discussed at the beginning of the introduction, yet one claims to be optimal and the other claims to be suboptimal. Clearly, only one of them can be right. But which one?

As we shall show, the answer to the above is itself obfuscated by the existing vague round counting methodology. In particular, the answer depends on whether or not the corresponding lower bounds render the last (send-only) rounds as necessary. As it turns out, the existing lower bound [DRS90], which uses fail-crashes only, does not render this round necessary. In fact, the lower bound does not require parties to wait to receive their $f + 2$ -round message. So does this mean that the above protocols are not round-optimal? Not necessarily! Since the lower bound holds for crash failures, it may be too loose to prove tightness for protocols tolerating Byzantine faults. A reader familiar with such round-complexity lower bounds of distributed fault-tolerant protocols might assume this is not the case, as most such tight bounds are in fact proven for weaker corruption/fault types and then matched by a protocol. Notwithstanding, in what we consider a surprising result, we are able to prove that considering Byzantine corruptions does add (at least) an extra receive-only round to the lower bound.

1.1.2 A Consistent Methodology for Counting (and Comparing) Rounds of Synchronous Protocols. A core thesis of our work is that the above ambiguity in claims is a result of a round-counting methodology, which is inadequate for crisply capturing the round-optimal esBA landscape. In order to untie the above Gordian knot, and devise a proper language that allows for a clear classification of different esBA protocols, we propose a fine-grained round-counting methodology, which explicitly disambiguates rounds in which parties need to send messages and still wait to receive messages sent to them from rounds in which they simply send messages and terminate. In a nutshell, every round is split into two mini-rounds: in the first mini-round, the party sends a message and might perform some computation on its local state; in the second mini-round, each party receives all messages sent to it in the first mini-round and updates its local state accordingly. When mapping this model to a real-world implementation of synchrony, the network latency (aka timeout) Δ is the time that passes between the first and second mini-rounds.

We refer to rounds in which a party needs to process both the first and the second mini-rounds as *send/receive rounds* (or, simply, *receive rounds*) and denote a protocol that requires t such rounds as a $t(r)$ -round protocol. We refer to a round in which the protocol only executes the first mini-round (and then terminates) as a *send(-only) round*, and denote a protocol that requires $t - 1$ send/receive rounds and an additional send-only round as a $t(s)$ -round protocol. Given the above round-counting methodology, it is straightforward to classify the aforementioned results: [BG93] and [ELP25] are $\min\{f + 3(s), t + 1(r)\}$ -round Consensus and Broadcast protocols, respectively, tolerating Byzantine parties. On the other hand, [PR04] and [PT86] are $\min\{f + 2(r), t + 1(r)\}$ -round Consensus and Broadcast protocols, respectively, tolerating omission faults. The known lower bound for BA [DRS90] assumes fail-crashes and requires at least $\min\{f + 2(s), t + 1(r)\}$ rounds.

1.1.3 Revisiting and Charting the Feasibility Landscape. Armed with such an accurate methodology for counting rounds, we can now investigate the gaps in the literature.

esBA in the Presence of Omission Faults. The first gap is with respect to omission corruptions. Recall that the existing lower bound is for fail-crashes and mandates $\min\{f + 2(s), t + 1(r)\}$ rounds. However, to our knowledge, the best known construction tolerating omission faults is by Perry and Tueg [PT86] and is a $\min\{f + 2(r), t + 1(r)\}$ -round Broadcast protocol (i.e., in case $f < t - 1$, the protocol from [PT86]

needs an additional receive mini-round which in terms of time incurs an extra latency Δ delay). We close this gap by constructing the first $\min\{f + 2(s), t + 1(r)\}$ -round Broadcast protocol tolerating any $t < n$ omission-corrupted parties. (We note in passing that our protocol follows the classical definition in [PT86] for Broadcast in the presence of omission failures, where parties output either a value or $\perp$.) By having every party broadcast his value in parallel and then adopt a value if it appears on the output more than $n/2$ times (otherwise output $\perp$) this also yields the first $\min\{f + 2(s), t + 1(r)\}$ -round Consensus protocol against omission-corrupted minorities. We note in passing that the $\min\{f + 2(r), t + 1(r)\}$ -round protocol in [PR04] was designed for the so called *uniform Consensus* problem (see Section B.3). Since this is a stronger variant of Consensus, it is conceivable that the lower bound is also tighter.

Tighter Lower Bounds. As discussed, the existing $\min\{f + 2(s), t + 1(r)\}$ lower bound (proven for crash faults) [DRS90] is not sufficient for proving tightness of the various existing esBA constructions that provide $\min\{f + 2(r), t + 1(r)\}$ -round BA protocols [BGP92, ZIP95] or $\min\{f + 3(s), t + 1(r)\}$ -round protocols [BG93, MW94, ELP25]. Filling in this gap we prove that $\min\{f + 2(r), t + 1(r)\}$ rounds are necessary for esBA tolerating Byzantine corruptions, thereby strengthening the lower bound of [DRS90] and establishing the optimality of [BGP92, ZIP95]. We further prove that for Broadcast tolerating dishonest (Byzantine) majorities, an extra send round is necessary, making $\min\{f + 3(s), t + 1(r)\}$ the lower bound for this case. Albeit existing $\min\{f + 3(s), t + 1(r)\}$ -round protocols are for honest (super-)majorities, as such their optimality remains an open question [BG93, MW94, ELP25].

Revisiting the EIG-based esBA Protocol [BG93]. Exponential Information Gathering (in short EIG) was introduced in the seminal paper of Lamport, Shostak, and Pease [LSP82] that introduced the problem of BA and gave first solution. Albeit, as the name suggests, EIG is an exponential space/time protocol—it effectively consists of multiple rounds in which the players relay all the information they have received to everyone, and then attempt to achieve a consistent decision (see Section 3.2 for details).

Similarly to the classical BA literature, where the first protocol was exponential (in the player set size n) time/space and was later on improved to polynomial (in fact, linear) [GM98], esBA experienced a similar trajectory. The first optimally resilient (tolerating up to $t < n/3$ corruptions) protocol was exponential [BGP92] and was in-fact relying on EIG. The follow-up work by Berman and Garay [BG93] achieved polynomial (in n) complexity by lowering the corruption tolerance to $t < n/4$. As discussed above, the work achieved (in our terminology) a $\min\{f + 3(s), t + 1(r)\}$ -round protocol. The high-level idea of [BG93] is to alternate between phases that use an EIG-style protocol for a few (constantly many) rounds, and phases where parties use smart voting to compress their state so that it remains polynomial. A similar approach was more recently taken by Abraham and Dolev [AD15] to provide an optimally-resilient (i.e., $t < n/3$) with a similar early stopping-bounds as [BG93].

Re-examining these results under the spectrum of our fine-grained round-counting exposes counter-intuitive results: As discussed, the lower bound we could prove for BA is $\min\{f + 2(r), t + 1(r)\}$ rounds (which is different from the $\min\{f + 3(s), t + 1(r)\}$ -round protocol of [BG93]). Notwithstanding we were able to revise the protocol of [BG93] so that it does not need to send the extra “done” message in the last (send-only) round. This modification requires non-trivial white-box surgery to the original protocol as outlined in Section 2.4. As we prove, the revised protocol does achieve the $\min\{f + 2(r), t + 1(r)\}$ bound which we prove optimal for $t < n/4$ Byzantine corruptions, restoring the optimality claim of [BG93].

A New Optimally Resilient Tight-Rounds Early-Stopping BA Protocol. Last but not least, we turn to the question of round-optimal and optimally-resilient esBA assuming a (signatures) setup. Here, Elsheimy, Loss, and Papamanthou [ELP25] recently provided an exponential early-stopping Broadcast protocol that makes use of ideal signatures, tolerates up to $t < n/2$ corruptions and terminates in $\min\{f + 3(s), t + 1(r)\}$ rounds. Recall that since we are in the $t < n/2$ realm, this trivially implies an exponential early-stopping Consensus protocol (for the bound $t < n/2$ which is optimal for Consensus) by having every party broadcast their input in parallel and take the majority of the outputs. However, as the authors observe, the protocol is not tight given the state-of-the-art lower bound $\min\{f + 2(s), t + 1(r)\}$ from [DRS90] (for crash faults). In fact, even given our newly proven lower bounds of $\min\{f + 2(r), t + 1(r)\}$ for Byzantine corruptions

(Theorem 2), there remains a gap. As our last (and most) technical contribution, we devise a new exponential $\min\{f + 2(r), t + 1(r)\}$ -round protocol. Our protocol follows a very different approach than that of [ELP25] and both the construction and proof are technically challenging.

In addition to the non-trivial contribution, our protocol replaces the assumption of ideal signatures with that of a setup for information-theoretic signatures [HSZI00,SHZI02]. The reason for this choice is that the use of ideal signatures within an exponential-time protocol has attracted criticism from the cryptographic community as an overly abstract hybrid assumption: how can we choose the parameters so that no adversary polynomial in them can forge signatures of honest signers? Instead, in this work we use the concrete construction of information-theoretic, in-short i.t. (pseudo-)signatures from [IOZ14]. Such i.t. pseudo-signatures are setup hungry primitives (i.e., they assume a correlated-randomness setup which is polynomial to the longest message to be signed) but provably achieve unforgeability against unbounded adversaries as long as they are not transferred to too many different users. As such, although in the abstract protocol design one can treat them like an information-theoretic analog of existentially-unforgeable signatures [GMR88], their concrete invocation requires careful choice of the setup size and extra care so that they are not reused (even verified) more than their allowed number of times. To keep the protocol description modular, in Section 8 where we provide our optimal esBA construction and proof intuition, we ideally treat them as existentially-unforgeable signatures. A discussion of how to instantiate them with i.t. signatures (and in particular how to chose the parameters) so that the way they are used in the protocol does not compromise their security guarantees is given in Appendix A.

Table 1. Our results in perspective with early-stopping Byzantine Agreement protocols with claimed concrete (non-asymptotic) round complexity $\min\{f + c, t + 1\}$ for a constant c , and which tolerate $O(n)$ corruptions. $t(r)$ denotes that the protocol is a t (send/)receive round protocol and $t(s)$ denotes that the protocol has $t - 1$ (send/)receive rounds plus one send(-only) round. The color-coding shows which bounds (in terms of round complexity) are tight: green colored lines correspond to tight bounds (the tightness follows from our new theorems and not by existing lower bounds); orange colored lines indicate that the bound is not proven to be tight (no matching lower bound or better protocol); and red colored lines indicate that the bound is provably not tight. “IS” is short for ideal signatures, and “USS” is short for unconditionally secure signature setup (see Appendix A for more discussion).

Protocols	Failures	BA	Resilience	Claimed Rounds	Fine-Grained Rounds	Model	#Comm.
[PT86]	Omission	Broadcast	$t < n$	$\min\{f + 2, t + 1\}$	$\min\{f + 2(r), t + 1(r)\}$	Plain	Poly(n)
[BGP92]	Byzantine	Consensus	$t < n/3$	$\min\{f + 2, t + 1\}$	$\min\{f + 2(r), t + 1(r)\}$	Plain	Exp(n)
[BG93]	Byzantine	Consensus	$t < n/4$	$\min\{f + 2, t + 1\}$	$\min\{f + 3(s), t + 1(r)\}$ ^a	Plain	Poly(n)
[MW94]	Byzantine	Broadcast	$t < n/8$	$\min\{f + 2, t + 1\}$	$\min\{f + 3(s), t + 1(r)\}$	Plain	Poly(n)
[ZIP95]	Byzantine	Broadcast	$t < n/8$	$\min\{f + 2, t + 1\}$	$\min\{f + 2(r), t + 1(r)\}$	Plain	Poly(n)
[GM98]	Byzantine	Consensus	$t < n/3$	$\min\{f + 5, t + 1\}$	$\min\{f + 5(s), t + 1(r)\}$	Plain	Poly(n)
[PR04]	Omission	Consensus	$t < n/2$	$\min\{f + 2, t + 1\}$	$\min\{f + 2(r), t + 1(r)\}$ ^b	Plain	Poly(n)
[AD15]	Byzantine	Consensus	$t < n/3$	$\min\{f + 2, t + 1\}$	$\min\{f + 2(r), t + 1(r)\}$	Plain	Poly(n)
[ELP25]	Byzantine	Broadcast	$t < n/2$	$\min\{f + 3, t + 1\}$	$\min\{f + 3(s), t + 1(r)\}$	IS	Exp(n)
Section 6	Omission	Broadcast	$t < n$		$\min\{f + 2(s), t + 1(r)\}$	Plain	Poly(n)
Section 7 (revised BG93)	Byzantine	Consensus	$t < n/4$		$\min\{f + 2(r), t + 1(r)\}$	Plain	Poly(n)
Section 8	Byzantine	Broadcast	$t < n/2$		$\min\{f + 2(r), t + 1(r)\}$	USS	Exp(n)

^a The original description of [BG93] is a $\min\{f + 3(r), t + 1(r)\}$ -round protocol; however, by appropriate surgery we can reduce it to a $\min\{f + 3(s), t + 1(r)\}$ -round protocol (See Section 2.4.)

^b [PR04] provides a construction for so called *uniform Consensus*, a stronger variant of Consensus (see Section B.3). As such our construction from Section 6 does not necessarily render it non-tight. A lower bound for uniform Consensus is an open problem.

2 Technical Overview

In this section, we briefly overview the technical architecture of this work.

2.1 Receive-Round $f + 2$ Lower Bound for Byzantine Faults

The $f + 2$ lower bound for BA protocols was first established by Dolev, Reischuk, and Strong [DRS90]. In fact, from the arguments in [DRS90, AS24], it is only guaranteed that if an honest party p is allowed to terminate in round $f + 1$, then there exists another honest party q that eventually terminates with a distinct value, thereby violating consistency. Furthermore, the lower bound proof assumes the weakest adversaries, i.e., assumes only crash faults. Under our refined notion of rounds, this result implies an $f + 2(s)$ lower bound against crash faults. Yet, since all existing BA tolerating Byzantine faults require at least $f + 2(r)$, this naturally raises the question of the tight (lower) bound for Byzantine faults:

What is the exact lower bound for BA protocols in the Byzantine fault model? Does there exist an optimal protocol that attains it?

We prove the $f + 2(r)$ lower bound for Byzantine faults, thereby completing the $f + 2(s)$ lower bound for crashed faults by [DRS90]. To understand the high-level intuition, we first recall the (restated) proof by Abraham and Stern [AS24] for crash faults. Since the protocol satisfies validity, there must be a *crucial* party p^* such that—fixing the inputs of all other parties—changing the input of p^* , and hence its first-round message, can alter the decision value of an honest party. If p^* crashes after sending its message to some party p_i , then from p_i 's view no faulty behavior is detected and p_i will terminate with some value v . However, other honest parties may delay their termination since a failure of p^* is detected, and their final decision may differ from v given that p^* is crucial. The same reasoning applies inductively through round $f + 1$: if one crash-faulty party is detected per round from 1 to f , then no honest party p_{i_f} can safely terminate in round $f + 1$ when no crash fault is detected. This leads to the $f + 2(s)$ lower bound.

In the Byzantine fault setting, the crucial party p^* is more subtle since it can send conflicting messages to different honest parties. Specifically, p^* can send message m to honest party p_i in some round, inducing p_i to terminate with value v , while simultaneously sending a conflicting message m' to another honest party p_j and inducing p_j to terminate with $v' \neq v$. Since p^* appears honest from the views of both p_i and p_j , allowing them to terminate in this round—where no Byzantine behavior (including the crash fault) has been detected—would violate consistency. Therefore, honest parties must wait for one additional round of messages before termination, yielding the $f + 2(r)$ lower bound as claimed.

2.2 Send-Round $f + 3$ Lower Bound for Broadcast with Dishonest Majority

From the above discussion, we have seen that $f + 2(r)$ are necessary for BA to tolerate up to f faults, and that there exist tightly matching early-stopping protocols. However, the round complexity is also closely related to the corruption threshold: intuitively, achieving higher fault tolerance inherently increases the difficulty of protocol design. Indeed, one of the most technically challenging parts of our work is to construct an optimal-round Broadcast protocol tolerating $t < n/2$, as will be discussed in detail later. Once such a Broadcast protocol is available, a round-optimal Consensus protocol can be trivially obtained by running one instance of Broadcast for each party. In this setting, $t < n/2$ is the best possible resilience bound we can hope for in Consensus. In contrast, Broadcast can tolerate a dishonest majority, but to our best knowledge no known protocol achieves termination within $f + c$ rounds for any constant c in that regime.

We further explore the lower-bound landscape of Byzantine Broadcast when $t \geq n/2$, and show that $f + 2(r)$ are no longer sufficient. To understand the intuition, consider the simplest case: Can a Broadcast protocol safely terminate within two rounds when $f = 0$ but $t > n/2$? Suppose we examine two fully honest executions, A_1^0 and A_1^1 , that differ only in the sender's input value. If the protocol terminates within two rounds (i.e., in receive-round 2), then under the assumption of a dishonest majority we can construct two additional executions, A_2^0 and A_2^1 , where half of the honest parties in A_2^i have a view identical to that in A_1^i for $i \in \{0, 1\}$, while the other half receive contradictory messages from Byzantine parties behaving as in A_1^{1-i} . Since the first half parties, which terminate in round 2 in A_1^i , cannot distinguish A_1^i from A_2^i at the end of round 2, they must also terminate in round 2 in A_2^i . The remaining honest parties—those observing inconsistent messages—may continue for extra rounds, but they are required to output the same value as the already-terminated honest parties. Finally, we consider a further execution, A_3 , where half of the honest

parties share the same round-2 view as in A_2^0 and the other half as in A_2^1 . Furthermore, other Byzantine parties become silent from round 3 as if they are in A_1^i , forcing other honest parties to make their outputs inconsistent and hence violating consistency.

This observation can be generalized to any f such that $f \leq 2t - n - 1$, where the first $2t - n - 1$ rounds are built via the technique used in the proof of $f + 2(r)$ lower bound—by making a single crash fault in each round. After $2t - n - 1$ rounds, there exists a crucial party that can determine the outputs by its state. Then, by regarding the crucial party as a sender in the $f = 0$ example and observing similar executions, we can extend the $f + 3(s)$ lower bound result to $f \leq 2t - n - 1$,

2.3 $f + 2(s)$ Broadcast Protocol Tolerating Omission Faults

While we have proven that $f + 2(r)$ are necessary for BA against Byzantine faults, it leaves another open question on the tightness of BAs against crash and omission faults. Is $f + 2(r)$ necessary or $f + 2(s)$ sufficient? It is somewhat surprising that, to the best of our knowledge, there is no known tight BA protocol tolerating crash or omission faults but existing ones require $f + 2(r)$ [PT86, PR04]. Hence, we fill the gap by providing a simple but elegant Broadcast protocol for any $t < n$, achieving $\min\{f + 2(s), t + 1(r)\}$ round termination. The protocol utilizes two special symbols, Null and $\perp$, to propagate any important misbehavior it observes in a certain round. Since there is no Byzantine behavior (but only crash and omission faults), once a party received a value $v \in \mathcal{D}$, which is not Null nor $\perp$, it can echo the value and terminate. The core problem is how to determine the safe condition to terminate with $\perp$, which implies that one can be sure that no other party terminates with another value. To help this decision, a party sends Null to others once it has not received any echoes of $v \in \mathcal{D}$ in a certain round. The Null messages constitutes the fact that the sender (of Null messages) is still honest to the receiver, although the sender could not receive any $v \in \mathcal{D}$ yet. Once the number of parties that have sent Null is larger than $n - r + 1$, the protocol guarantees that there will be no one terminating with a value other than $\perp$.

2.4 Revisiting BG93

We observe that Cloture Vote protocol in [BG93] requires $\min\{f + 3(s), t + 1(r)\}$ in our round counting model, since a party is required to send “I am done” after it outputs a value in (at most) round $f + 2(r)$. At a high level, the consistency of the Debate protocol relies on *Liveness*: if an honest party p resolves a node σ in round r , then all other honest parties must also resolve σ consistently by round $r + 1$. Here $\sigma = p_1 \dots p_\ell$ denotes a sequence of (non-repeating) parties indexing a node in the EIG tree structure (see Section 3). To achieve this property, p must send messages in round $r + 1$, though this requirement is not explicitly stated in [BG93]. Again, in the worst case, honest parties may decide their output value in some Debate protocol in round $f + 2$, which forces them to send the corresponding messages in round $f + 3$ even though no honest party needs to receive them.

We propose a revised version of [BG93] that reduces the round complexity to $\min\{f + 2(r), t + 1(r)\}$ by the following modifications. First, we switch the meaning of 1 and 0 for ‘cloture’ in the invocation of the Debate protocol. In the Cloture Votes protocol in [BG93], the “I am done” message from p is indeed unnecessary, as it merely indicates that p will vote ‘Continuation’ permanently. If silence is simply interpreted as ‘Continuation’ at the beginning of each Debate, then the “I am done” message becomes redundant. Therefore, in our revised Cloture Votes protocol (Fig. 8), we redefine 0 to represent ‘Continuation’ and 1 to represent ‘Cloture’, since the default value for silence in Debate is 0. Second, we introduce a new rule for handling silence in the Debate protocol: If no message is received from σx , then the value of σx is assumed to be the same as that of σ . This rule effectively ensures Liveness although there is no message from honest parties that have already resolved the node in the previous round. Naturally, no party can resolve a node before Debate starts. Therefore, if the first message in Debate is missed, then it is replaced by 0, consistent with the rule above. With this modification, honest parties no longer need to send messages about nodes resolved in the current round. In addition, we present more detailed proofs and specification of the early-stopping variant in [BG93] to help readers better understand this seminal work.

2.5 Early-Stopping Broadcast for $t < n/2$

The construction of $f + 2(r)$ -round early-stopping Broadcast is challenging. Toward this, an honest party p must terminate in some round r that appears to be round $f + 2$ from p 's perspective. However, r may not actually be the round $f + 2$, as Byzantine parties can deceive p into believing that $r = f + 2$. Once p terminates, it can no longer send information, and all other honest parties must still terminate with a decision value consistent with p . If t is sufficiently small, Byzantine parties must involve a large number of honest parties to deceive p into believing that $r = f + 2$. In this case, the remaining honest parties can easily infer and follow p 's decision value without further communication from p . However, when t is large, the problem becomes significantly more difficult. In such cases, the other honest parties must treat p 's silence itself as a source of information.

[BGP92] is the first protocol to demonstrate how to exploit silence to achieve early stopping by round $f + 2$. The silence of p indicates that p has already resolved the ancestor node and therefore requires no further information about that branch. (If the ancestor node is the root, this means that p has terminated.) Thus, silence must not be treated as the receipt of a default message, since it conveys the information about the specific value p has resolved. This idea is further simplified in [AD15]: If no message is received from p regarding σ , the value from σ is used instead of a default value. Using this approach, [BGP92] and [AD15] achieve early stopping by round $f + 2$ under the condition $t < n/3$.

When $t \geq n/3$, Byzantine Broadcast protocols additionally rely on signatures for authenticity: Every message sent by a party is accompanied by a signature, allowing others to publicly verify its origin. Moreover, the message itself may include all messages and signatures from previous rounds, forming a *signature chain* that possibly originated from the designated sender. However, this situation makes another challenge to achieve early stopping by round $f + 2$. As discussed above, if a party p does not send a message with respect to a node in the EIG tree, then it indicates that p has already resolved an ancestor node of it. However, this also breaks the continuity of the signature chain. Namely, if silence represents that an ancestor node is resolved, then a party q must not remain silent when q fails to receive a message from p ; otherwise, q 's silence would incorrectly suggest that it has resolved an ancestor node, when in fact it has not. At the same time, q cannot produce a valid message continuing from p , due to the lack of the valid signature from p . To address this, q must instead generate a special message indicating ‘I did not receive a message from p ’. In our protocol, we denote this with the symbol Null. Nevertheless, it is possible that a Byzantine party x generates a Null message and q fails to receive that message. In this case, q must generate Null message to indicate that it did not receive x 's message. Importantly, q 's Null message is not an echo of x 's message, and the two messages are distinguished. Hence, it is crucial that a Null message explicitly specifies **who** missed **whose** message.

Our protocol allows parties to start new signature chains dynamically during execution. In the traditional EIG Broadcast protocol with signatures, only the signature chain initiated by the sender in round 1 is considered valid [DS83,ELP25]. In contrast, our protocol permits the signature chain of a Null message to be initiated by any party in any round. This capability ensures that honest parties can correctly convey information about **who** missed **whose** message. The intuitive description of the protocol is described in Section 8.3.

3 Preliminaries

For integers $a \geq b \geq 1$, $(a)_b$ denotes a falling factorial, i.e., $(a)_b = a(a-1) \dots (a-b+1)$. Define $(a)_0 = 1$. For $a < b$, $[a..b]$ is the set containing all integers between a and b , i.e., $[a..b] := \{a, a+1, \dots, b-1, b\}$. $\mathcal{D}$ denotes a set of possible input values.

Throughout the paper, we use $\mathcal{P} = \{p_1, \dots, p_n\}$ to denote the set of all n parties. Let $t \leq n$ denote the upper bound on the number of faulty parties, and $f \leq t$ denote the actual number of faulty parties. We use $[\mathcal{P}]^*$ to denote the set of all finite sequences over symbols in $\mathcal{P}$ but *without repetition*. For example, if $\mathcal{P} = \{a, b\}$, then $[\mathcal{P}]^* = \{\varepsilon, a, b, ab, ba\}$. We further define $[\mathcal{P}]^\ell$ as the set containing all length ℓ sequences *without repetition* in $[\mathcal{P}]^*$. For example, if $\mathcal{P} = \{a, b, c\}$, then $[\mathcal{P}]^0 = \{\varepsilon\}$, $[\mathcal{P}]^2 = \{ab, ac, ba, bc, ca, cb\}$, and

$[\mathcal{P}]^* := \bigcup_{i=0}^{\infty} [\mathcal{P}]^i = \bigcup_{i=0}^3 [\mathcal{P}]^i$. For $\sigma \in [\mathcal{P}]^*$, we use $[\sigma]$ to denote the set of all characters appearing in σ , e.g., if $\sigma = abc$, then $[\sigma] = \{a, b, c\}$.

3.1 Problem Setting and Assumptions

Network and Synchronous Model. We consider a fully connected network of n parties $\mathcal{P}$ executing a protocol in a synchronous setting. All parties share a global clock, and there exists a known upper bound Δ on the message delay between honest parties. This assumption allows each protocol execution to proceed in discrete *rounds* of length Δ , where every message sent by an honest party at the beginning of round r is guaranteed to be delivered to all honest parties by the end of that round (or, by the beginning of round $r + 1$).

Following the structure required for our analysis, we further refine each round into two mini-rounds: a *send* (mini-)round and a *receive* (mini-)round. Note that this decomposition is purely syntactic and does not modify the underlying synchrony assumption or the known delay bound Δ . We discuss the new model in more details in Section 4. From here, we always consider the round model in which each round starts with the send-round, and all messages sent in round r are received by round r .

Adversarial Model. We assume an adversary (either probabilistic polynomial-time (PPT) or computationally unbounded) that can adaptively corrupt up to t parties, where t can be varied with respect to the protocols we will discuss. Once corrupted, a party is under full control of the adversary and may deviate arbitrarily from the prescribed protocol, including sending inconsistent or forged messages. The adversary can choose which parties to corrupt during the execution, based on the entire protocol transcript observed so far. Unless otherwise specified, an adversary is rushing—it can observe all messages sent by honest parties within a round before determining the messages of corrupted parties for the same round.

All honest parties are assumed to know the parameters (n, t, Δ) in advance and to have authenticated point-to-point channels between every pair of parties.

Signatures. In Section 8 we utilize (digital) signatures [GMR88] to ensure authenticity, non-repudiation, and transferability. To better focus on the protocol and proofs, we skip the setup of the signature scheme, and simply assume all parties have access to ideal signing and verification in a black-box way with the following properties:

- *Correctness:* Signatures generated by honest parties always verify.
- *Unforgeability:* Except with negligible probability,⁵ no adversary can generate a valid signature of an honest party on a fresh message.
- *Consistency:* Except with negligible probability, no adversary can create a signature that verifies differently for two (honest) parties.

We emphasize that our construction requires information-theoretic signatures [HSZI00, SHZI02, SS11], i.e., the above properties hold even against computationally unbounded adversaries. This requirement stems from that a party (no matter honest or Byzantine) is assumed to run in exponential time, since the message size (and thus the signing and verification time) increases exponentially with n . To maintain security in this setting, we resort to the information-theoretic signatures, albeit at the cost of an exponentially expensive setup phase. In Appendix A, we provide the formal definitions of information-theoretic signatures along with a detailed asymptotic complexity analysis of our protocol in Section 8.

Byzantine Agreement Problems. Now we provide formal definitions of Byzantine Broadcast and Byzantine Consensus. In this work, we collectively refer to these as the Byzantine Agreement (BA).

Definition 1 (Byzantine Broadcast). *Let Π be a protocol executed among n parties $s, p_1, \dots, p_{n-1}$, where a designated sender s holds initial input $v \in \{0, 1\}$ and all parties p_i output a value $y_i \in \{0, 1\}$ when terminating. The protocol Π achieves (t -)Byzantine Broadcast, if the following properties hold even when at most $t < n$ parties are corrupted.*

⁵ More precisely, unforgeability and consistency hold except with a probability negligible in a given parameter λ , which might be as large as $O(2^n)$. See Appendix A for a detailed analysis on the signature setup.

- **Validity.** If the sender s is honest and has input v , then all honest parties output $y_i = v$.
- **Consistency (Agreement).** All honest parties output the same value.
- **Termination.** All honest parties terminate and output.

Definition 2 (Byzantine Consensus). Let Π be a protocol executed among n parties $p_1, \dots, p_n$, where each party p_i holds initial input $v_i \in \{0, 1\}$ ⁶ and outputs a value $y_i \in \{0, 1\}$ when terminating. The protocol Π achieves (t) -Byzantine Consensus, if the following properties hold even when at most $t < n/2$ parties are corrupted.

- **Validity.** If all honest parties have initial input v , then all honest parties output $y_i = v$.
- **Consistency (Agreement).** All honest parties output the same value.
- **Termination.** All honest parties terminate and output.

3.2 Exponential Information Gathering (EIG)

The celebrated work of EIG (Exponential Information Gathering) solves BA with exponential complexity [PSL80,BNDDS92]. The concept of EIG protocol (for both Broadcast and Consensus) is later abstracted as an EIG tree as a data structure. Let $\mathcal{P} = \{p_1, p_2, \dots, p_n\}$ be the assemble of n parties and $t < n/3$ be the upper bound of corrupted parties. During the protocol every party p maintains an EIG tree $\mathcal{IT}_p$ of the following properties.

- The height of $\mathcal{IT}_p$ (in the basic EIG protocol [PSL80,BNDDS92]) is $t + 1$ for the Broadcast problem and $t + 2$ for the Consensus problem (including additionally the 0-th level). Note that the height may be smaller in early-stopping variants.
- In the broadcast problem, the root in the first level of $\mathcal{IT}_p$ is indexed by the sender s and stores the initial value of s as received by p . In the Consensus problem, the root in the 0-th level⁷ is indexed by an empty sequence ε . It has n child nodes, each indexed by a party, storing that party's initial value as received by p .
- A node in the ℓ -th level of $\mathcal{IT}_p$ is indexed by a sequence $\sigma = p_{i_1}p_{i_2} \dots p_{i_\ell}$ for $\ell > 0$ (or $\sigma = \varepsilon$ for $\ell = 0$), where σ consists of ℓ distinct parties in $\mathcal{P}$ (including p itself). The value $\mathcal{IT}_p(\sigma)$ at this node represents the statement:

“ p_{i_ℓ} said that $p_{i_{\ell-1}}$ said that ... that p_{i_2} said that p_{i_1} said $\mathcal{IT}_p(\sigma)$.”

- A node $\sigma = p_{i_1}p_{i_2} \dots p_{i_\ell}$ has $n - \ell$ child nodes, each indexed by $p_{i_1}p_{i_2} \dots p_{i_\ell}p'$ for $p' \in \mathcal{P} \setminus [\sigma]$, i.e., a party that is not included in σ .

The protocol proceeds in $t + 1$ rounds. In the first round, every party sends its input value to all others, and each party stores the value in the corresponding node of its EIG tree. Then, in every subsequent round r , each party generates a message containing all the $r - 1$ -level nodes of their IT tree and sends it to the others. Upon receiving the messages, p sets the value of node σq in $\mathcal{IT}_p$ to the value that q claims to store at node σ in $\mathcal{IT}_q$. After $t + 1$ rounds, p builds a resolution tree $\mathcal{RT}_p$ based on the following *Resolve Rules* and finally outputs the root value of $\mathcal{RT}_p$.

$$\mathcal{RT}_p(\sigma) = \begin{cases} \mathcal{IT}_p(\sigma) & \text{Leaf rule: if } \sigma \text{ is a leaf node,} \\ v & \text{Majority rule: if the majority of } \mathcal{RT}_p(\sigma q) \text{ is } v, \\ v_0 \text{ (default value)} & \text{Tie-breaker rule: otherwise.} \end{cases}$$

Note that the Resolve Rules are applied to the nodes in a bottom-up manner. We present a toy example of an execution of the EIG protocol in Fig. 1, depicting the views $\mathcal{IT}_{p_1}$ and $\mathcal{RT}_{p_1}$ of an honest party p_1 .

When the executing party p is clear from the context, we simply write $\mathcal{IT}(\sigma)$ and $\mathcal{RT}(\sigma)$. For each σ , we assume that both $\mathcal{IT}(\sigma)$ and $\mathcal{RT}(\sigma)$ are initialized to $\perp$.

⁶ A more general definition does not restrict the input and output to be binary values, yet it remains equivalent to the definition here.

⁷ We introduce the 0-th level for the EIG tree in the Consensus problem, to keep notions simple and compatible with the Broadcast problem.

Theorem 1 ([BNDDS92]). *The EIG protocol solves the BA problems provided that $n > 3t$.*

Although the EIG protocol achieves BA, it suffers from a significant drawback: exponential message complexity $O(n^t)$ and exponential running time. Consequently, considerable effort has been devoted to improving its complexity. A key observation is that a party p can often determine the value of $\mathcal{RT}_p(\sigma)$ in the EIG protocol before round $t + 1$. Once p fixes the resolved value of σ , it can stop gathering information about the entire subtree of $\mathcal{IT}_p$ rooted at σ , hence reducing the message complexity. This phenomenon is commonly referred to as *fixing* or *prediction* [MW94,BG93,GM98].

In the original EIG protocol, nodes are resolved at the end of execution based on the values contained in $\mathcal{RT}$. However, nodes can instead be resolved earlier during the execution using the values contained in either $\mathcal{IT}$ or $\mathcal{RT}$. We classify resolve rules based on $\mathcal{IT}$ as *IT resolve rules*, and those based on $\mathcal{RT}$ as *RT resolve rules*. The details of resolve rules depend on the specific protocol.

Another important technique is *fault masking*. During protocol execution, a Byzantine party q may attempt to prevent nodes from being resolved by, for example, sending inconsistent messages. However, such behavior also provides evidence that q is Byzantine. Once q is detected as Byzantine, honest parties can simply ignore the messages from it. One particularly simple form of fault masking is to treat a party detected as Byzantine as sending a fixed value (e.g., 0) for all nodes of interest. Therefore, if all honest parties recognize that q is Byzantine (we refer to this state as *globally detected*), it is equivalent to assuming that q sends the same message to all honest parties, implying that q can no longer disrupt the agreement process. This technique has been utilized in various works to reduce the message complexity [DRS90,BNDDS92,MW94,BG93,GM98,AD15].

In spite of these techniques, EIG methods still suffer from exponential message complexity. Several EIG protocols [BG93,GM98,AD15] employ a mechanism known as the *monitor vote* to attain polynomial message complexity. The monitor vote process serves as a mechanism to ensure the safe abort of the EIG protocol when the data size of honest parties becomes excessively large. This concept was first introduced by Berman and Garay [BG93], under the name *Cloture Votes* protocol.

[BG93] first introduced the *Debate* protocol, an EIG-based protocol with a round complexity of $\min\{f + 2, t + 1\}$ and resilience up to $t < n/4$.⁸ However, the Debate protocol suffers from exponential message complexity. To achieve a polynomial message complexity, the Cloture Vote protocol recursively invokes the Debate protocol in each round. These subprotocols (Debate protocols) are identical to the Debate protocol invoked in round 1, except that each is initiated in a later round, with the parties' inputs initialized to a special symbol 'Continuation'. More precisely, if the data size of an honest party p becomes excessively large, i.e., beyond a predefined threshold, then p votes for 'Cloture'; otherwise, it votes for 'Continuation' in that round.

Since the Debate protocol itself is an early-stopping BA protocol, it can be used to reach a consistent decision on whether to cloture, thereby ensuring agreement among all honest parties. Furthermore, the Debate protocol guarantees that at least one Byzantine party is excluded from the protocol in each round (otherwise, it terminates within a constant number of rounds). As a result, when a subprotocol is invoked in round r , at least $r - 1$ Byzantine parties have already been excluded from that Debate. Therefore, the Debate protocol invoked in round r can also terminate by round $\min\{f + 2, t + 1\}$, even though it starts in round r .

The protocol proposed by Abraham and Dolev in [AD15] has a similar structure to that of [BG93]. They first introduce an EIG subprotocol (like the Debate protocol in [BG93] but without a designated name), which achieves BA with a round complexity of $\min\{f + 2, t + 1\}$ and resilience up to $t < n/3$, but suffers from exponential message complexity. To obtain polynomial message complexity, the Monitor Vote protocol (the main protocol) repeatedly invokes the EIG subprotocol in each round. However, the difference in resilience required a much more complicated and sophisticated approach.

⁸ Although the authors claim that the protocol achieves early stopping within $\min\{f + 2, t + 1\}$ rounds, the original description actually requires $\min\{f + 3(r), t + 1(r)\}$ but the bound can be easily improved to be $\min\{f + 3(s), t + 1(r)\}$.

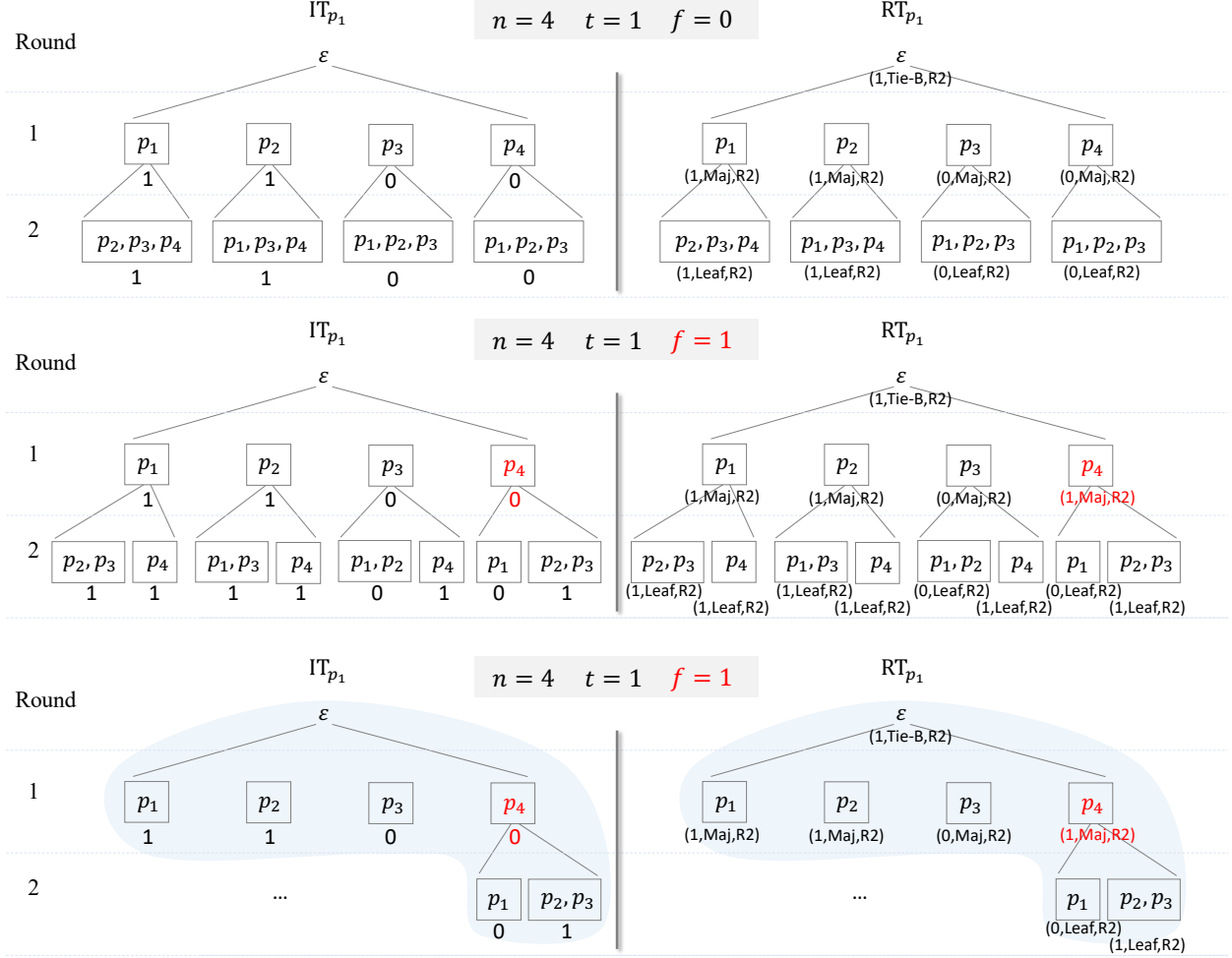


Fig. 1. Examples of the EIG protocol. **Top:** Views of $\mathcal{IT}_{p_1}$ and $\mathcal{RT}_{p_1}$ of honest party p_1 after an honest execution. **Middle:** Views of $\mathcal{IT}_{p_1}$ and $\mathcal{RT}_{p_1}$ of honest party p_1 where p_4 is Byzantine and the corresponding nodes are highlight in red. **Bottom:** the same as the middle, but only relevant branches are shown. Since all honest parties can consistently resolve the nodes p_1 , p_2 , and p_3 , their child nodes are irrelevant to the final output of the execution and hence are omitted in the bottom. On the other hand, since the resolved value of the node p_4 is determined by its child nodes, those child nodes are necessary to determine the output value. The value (X,Y,Z) of a node in $\mathcal{RT}$ presents that the value X is assigned according to rule Y by round Z , and “Leaf”, “Maj”, and “Tie-B” denote the leaf rule, the majority rule, and the tie-breaker rule, respectively.

3.3 Notation Conventions for BA Protocols

Let $\mathcal{P}$ be the set of parties with $|\mathcal{P}| = n$, and $\mathcal{D}$ denote a set of possible input values. Let $\mathcal{F}_p$ be the set of Byzantine parties detected by party $p \in \mathcal{P}$ when corresponding Byzantine detection rules are given. If p is clear from the context, we simply write $\mathcal{F}$ instead of $\mathcal{F}_p$. Similarly, we define $\mathcal{H}_p := \mathcal{P} \setminus \mathcal{F}_p$ and write $\mathcal{H} = \mathcal{H}_p$ when p is given. Note that $\mathcal{H}$ may contain Byzantine parties yet to be detected. We use lowercase letters to denote parties in $\mathcal{P}$. To ease understanding, we typically use p and q to denote parties playing a major role in the proof, and x, y, z for arbitrary parties that could be either honest or Byzantine.

The Greek letter σ denotes a sequence (possibly empty ε) as well as a node with the label σ in an EIG tree. Thus, we write σp for the child of node σ obtained by appending $p \in \mathcal{P}$ to the sequence σ . The quantity $|\sigma|$ denotes the length of the sequence σ . For a k -level node σ , it always holds that $|\sigma| = k$. We use the following terms in the proofs later: *reliable nodes* and *unreliable nodes*.

Definition 3 (Reliable Nodes and Unreliable Nodes). *A node represented as σ is called an unreliable node if no honest party is included in the set $[\sigma]$. Note that ε is also unreliable node. On the other hand, σp is called a reliable node if p is an honest party, while σ is an unreliable node.*

Message Format. Let $(\mathcal{P})^\ell$ denote the vector as an ordered $[\mathcal{P}]^\ell$ with a fixed canonical ordering. E.g., for $\mathcal{P} = \{p_1, p_2, p_3\}$ and $\ell = 2$,

$$(\{p_1, p_2, p_3\})^2 = (p_1 p_2, p_1 p_3, p_2 p_1, p_2 p_3, p_3 p_1, p_3 p_2).$$

$(\mathcal{P})^\ell[i]$ denotes the i -th component of $(\mathcal{P})^\ell$ for $i \in [(|\mathcal{P}|)^\ell]$. For example,

$$(\{p_1, p_2, p_3\})^2[1] = p_1 p_2, (\{p_1, p_2, p_3\})^2[2] = p_1 p_3, (\{p_1, p_2, p_3\})^2[3] = p_2 p_1.$$

Note that using the above notation, any $(\mathcal{P})^\ell$ can be written as

$$(\mathcal{P})^\ell = \left((\mathcal{P})^\ell[1], \dots, (\mathcal{P})^\ell[|\mathcal{P}|^\ell] \right).$$

Let $\ell_r = |(\mathcal{P} \setminus \{p\})^{r-1}|$. In round r , each party $p \in \mathcal{P}$ is assumed to send to all parties an ordered set

$$\widehat{\text{OUT}}_p^r := \left\{ ((\mathcal{P} \setminus \{p\})^{r-1}[1], p, v_{p,1}^r), \dots, ((\mathcal{P} \setminus \{p\})^{r-1}[\ell_r], p, v_{p,\ell_r}^r) \right\},$$

where $v_{p,1}^r, \dots, v_{p,\ell_r}^r$ are values determined by the specific protocol.

For a sequence $\sigma \in [\mathcal{P}]^*$, we define the σ -projection of $\widehat{\text{OUT}}_p^r$ as the ordered subset of $\widehat{\text{OUT}}_p^r$ that consists of elements indexed by sequences $(\mathcal{P} \setminus \{p\})^{r-1}[\cdot]$ with prefix σ . For example, if $\mathcal{P} = \{p_1, p_2, p_3, p_4\}$, then p_2 -projection of $\widehat{\text{OUT}}_{p_1}^3$ is $\{(p_2 p_3, p_1, v_{p_1,1}^3), (p_2 p_4, p_1, v_{p_1,2}^3)\}$. In later EIG protocols, instead of sending $\widehat{\text{OUT}}_p^r$, a party p is allowed to send only a part of $\widehat{\text{OUT}}_p^r$ by applying the prediction discussed in the previous section. If p has already closed the node σ before round r , then the σ -projection of $\widehat{\text{OUT}}_p^r$ can be excluded. As a result, we denote OUT_p^r as the message p should send by following a given protocol, and $\widehat{\text{OUT}}_p^r$ can be obtained by applying the prediction to OUT_p^r . The projection of OUT_p^r is defined naturally.

Many works in the literature informally describe the rule as “party p echoes all received messages.” More precisely, let $\mathcal{IN}_{q \rightarrow p}^{r-1}$ be the message (as a set of projections) the p receives from q in round $r-1$. Then the echo of $\mathcal{IN}_{q \rightarrow p}^{r-1}$ (w.r.t. p) denotes the ordered subset $E_{q \rightarrow p}^{r-1} := \{(\sigma_i q, p, v_i) | (\sigma_i, q, v_i) \in \mathcal{IN}_{q \rightarrow p}^{r-1}\}$.⁹ Consequently, p has $\text{OUT}_p^r = \bigcup_{q \in \mathcal{P} \setminus \{p\}} E_{q \rightarrow p}^{r-1}$ as its “echoes to all received messages”.¹⁰

⁹ If signatures are used then the echo may also contain a signature by p .

¹⁰ In the protocols considered in this work, whenever an honest party p sends a message, we also regard p as receiving that message since some resolve rules rely on this convention. However, as defined here, this self-reception is not taken into account when p generates the message to be sent in the next round.

4 A Refined Model to Count Rounds

In the synchronous setting, time is partitioned into discrete *rounds* of fixed duration Δ . All honest messages sent within a round are guaranteed to be delivered to their honest recipients within Δ time from their send event. This abstraction lets us measure time complexity in integral round units, independent of variations inside the round.

Two equivalent but notationally different conventions are common. In the *send-first* convention (left in Fig. 2), a round r begins with parties sending and ends when those messages are delivered. In the *receive-first* convention (right in Fig. 2), a round r begins with parties receiving the messages sent in round $r - 1$, followed by local computation and sending for round r . Although these conventions differ only by an index shift, they create ambiguity when comparing protocols or composing subprotocols—e.g., one protocol may terminate “after sending”, another “after receiving”.

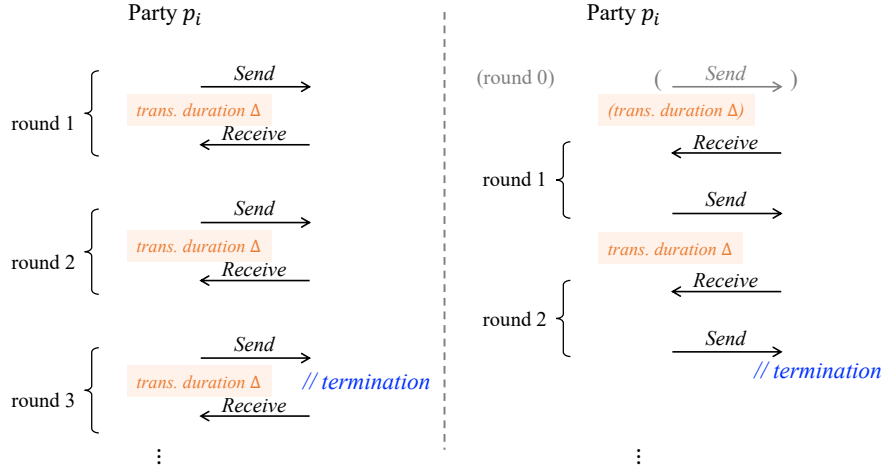


Fig. 2. Two synchronous models for counting rounds in the literature: **Left:** send-first model (e.g., [DRS90,BGP92,GM98,CBS04,PR04,AD15,FLZL21,ELP25]), and **Right:** receive-first model (e.g., [ZIP95,ACD⁺19,KW20]). Note that there might be a round 0 where the party p_i receives the initial input or the first protocol message from a particular sender. The **termination** at a specific moment is regarded as termination by round 3 on the left, and by round 2 on the right.

To remove this ambiguity, we refine each round into two conceptual *mini-rounds*: a *send-(mini-)round* followed by a *receive-(mini-)round*. During the send-mini-round of round r , each party computes and transmits its messages based on its current state. During the receive-mini-round of the same round, each party receives all messages that honest parties sent in send-mini-round r and updates its state accordingly. This refinement does not change the synchrony assumption; it simply fixes a uniform boundary for stating progression and termination. We allow computation within each mini-round, i.e., each send/receive-round starts with sending/receiving messages, followed by local computation, and then jumps to the next mini-round.

In this work, we primarily adopt the send-first convention. The separation into send- and receive-mini-rounds, as well as all our results, naturally extend to the receive-first convention.

Definition 4 (Mini-Round Decomposition). A synchronous round r consists of two ordered mini-rounds:

1. **Send-(mini-)round:** each party executes its send algorithm and transmits its messages;
2. **Receive-(mini-)round:** all messages sent by honest parties in round r are delivered and processed; each party updates its local state.

A protocol may specify its termination point as occurring after either the send- or the receive-mini-round of the final round.

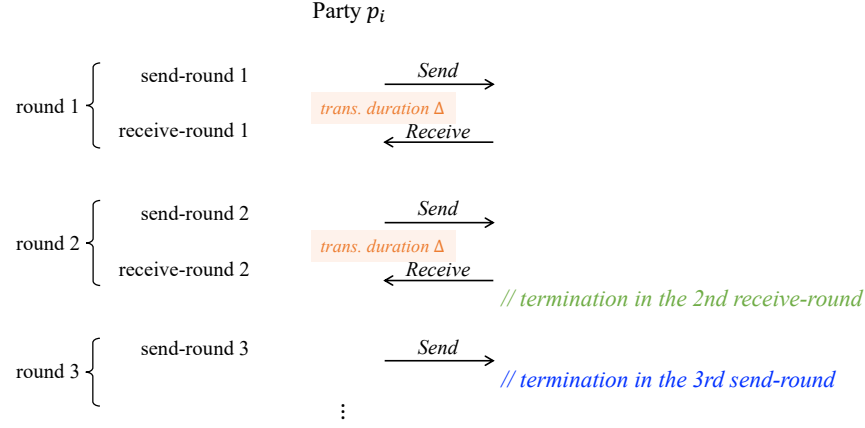


Fig. 3. Our refined synchronous model, where a protocol starts with a send-round.

Definition 5 (Termination Round). A protocol is said to terminate in the send-round r if it halts after completing the r -th send-round and before the r -th receive-round. It terminates in the receive-round r if it halts after completing the r -th receive-round and before the $(r + 1)$ -th send-round.

In this paper we assume a protocol always starts with a send-round, and therefore the following facts hold:

1. If $r > 1$, then a protocol that terminates in the r -th send-round always needs $r - 1$ receive-rounds.
2. If $0 \leq f \leq t$ and $x \in \mathbb{N}^+$, then $\min\{f + x(s), t + 1(r)\}$ refers to
 - send-round $f + x$ if $f + x \leq t + 1$,
 - receive-round $t + 1$ if $f + x > t + 1$.
3. If $0 \leq f \leq t \leq n$ and $x \in \mathbb{N}^+$, then $\min\{f + x(s), t + 1(r), n - 1(r)\}$ refers to
 - send-round $f + x$ if $f + x \leq t + 1$ and $f + x \leq n - 1$,
 - receive-round $t + 1$ if $f + x > t + 1$ and $t \leq n - 2$,
 - receive-round $n - 1$ if $f + x > n - 1$ and $t > n - 2$.

5 Revisiting the Lower Bounds

In this section, we revisit the $f + 2$ lower bound for crash faults in [DRS90], and provide two tighter lower bounds, i.e., the $\min\{f + 2(r), t + 1(r)\}$ lower bound for Byzantine faults and the $\min\{f + 3(s), t + 1(r)\}$ lower bound for Broadcast with dishonest majority, under our refined model in Section 4.

5.1 $f + 2(r)$ Lower Bound for Byzantine Faults

Dolev, Reischuk, and Raymond [DRS90] proved the $f + 2$ lower bound—more precisely, $f + 2(s)$ lower bound—for (eventual) BA protocols. In this section, we follow the proof by Abraham and Stern [AS24] for crash faults, and devise a new impossibility result, i.e., the $f + 2(r)$ lower bound, for BA protocols with Byzantine faults distinguished from those for crash faults.

Recall that for the crash fault, the (faulty) party will either send an honest message, uniquely determined by the protocol with respect to each receiver, or crash, i.e., it will send no message at all. However, faulty parties with Byzantine behaviors can send conflicting messages to different parties. If there exist multiple different faulty behaviors, the protocol may terminate with different outputs for each behavior. This is a crucial distinction when we first try to prove an impossibility result for Byzantine faults instead of crash faults. However, we will show that crash fault behavior is sufficient to generate critical configuration, and only one Byzantine behavior in critical round is enough to result in inconsistency: There is at most one faulty party per round, and the faulty party follows the protocol description, except in the following cases: 1) it may crash; and 2) when generating a message to send, it may compute the message as if it had not received a message from the previous faulty party. This limited behavior is sufficient to induce inconsistency by round $f + 1$.

If a party sends conflicting messages to honest parties, then the honest parties may detect the faulty behavior and ignore the faulty party in subsequent rounds. Building on this observation, to prove Theorem 2, we assume that the Byzantine party crashes at the end of the round in which it behaves in a Byzantine manner. This assumption simplifies our analysis without affecting the tightness of the bound.

In later rounds, the adversarial strategy can be more complex, but somewhat surprisingly, our simplification can show a tight bound for some cases, which implies that the behavior we described is indeed the best strategy for adversaries in some cases. For warm-up, we start by showing that $f + 2(r)$ are necessary for BA, where we will consider only two possible messages for each Byzantine faults.

Definition 6 (Configuration). *A configuration of a BA protocol consists of the contents of all parties at a particular point of the execution, which includes a party's initial values and current states if it is not terminated or crashed yet, or a party's output if it has terminated or crashed (in which the output is $\perp$).*

Since we assume that a Byzantine party crashes at the end of the round in which it behaves maliciously, the configuration remains well-defined even in the presence of Byzantine parties. Note that this is not generally true for arbitrary Byzantine behavior, as it is meaningless to reason about a party's internal state once it becomes corrupted.

Definition 7 (Almost Same Configurations [AS24]). *Two configurations are Almost Same (AS) if the states of all parties except for a single party are the same.*

Example. Consider a configuration C at the end of round r such that both p_i and p_j are not terminated, and p_i sends m to p_j in round r and then crashes. Define configuration C' , which is identical to C except that p_i sends $m' \neq m$ to p_j (which includes the case that p_i sends nothing to p_j) in round r . Then C and C' are AS since all states except for p_j are the same. Note that p_i is crashed at the end of round r and the state keeps identical in both C and C' . $\triangleleft$

Definition 8 (Failure Free Different Configurations [AS24]). *Two configurations are Failure Free Different (FFD) if the failure free continuation of one configuration leads to a different decision value than the failure free continuation of the other configuration.*

Remark 1. The original definition of FFD in [AS24] considers only crash faults, so the notion of a failure-free continuation can be naturally defined. In contrast, when considering Byzantine faults, it becomes unclear how to define the state of a Byzantine party at the end of a round in which it sent two contradictory messages, i.e., messages that could only have been generated from two distinct internal states. While defining a failure-free continuation under Byzantine faults is challenging,¹¹ we circumvent this difficulty by restricting the adversary: we simply assume that a Byzantine party crashes at the end of the round in which it misbehaves, as explained above.

This idea effectively helps us prove the $f + 2(r)$ lower bound, in a manner similar to how the $f + 2(s)$ lower bound is established for crash faults [DRS90, AS24]. Nevertheless, to prove the $f + 3(s)$ lower bound

¹¹ More generally, one could consider a trivial continuation of a given configuration, which we do not address in this work.

in the dishonest majority setting, which we present in the next section, we needed to consider a stronger adversary. Still, the restricted adversary model suffices for this result.

Theorem 2. *Suppose $0 \leq f \leq t \leq n - 1$, then any BA protocol must have an execution with receive-round $\min\{f + 2, t + 1, n - 1\}$.*

Before proving Theorem 2, we first claim the following three lemmas. Note that we consider Byzantine agreement in the proofs, but they all hold for Byzantine broadcast.

Lemma 1 (Lemma 1 in [AS24]). *Any BA protocol that is resilient to at least one Byzantine party must have two initial configurations that are AS but FFD.*

Proof (of Lemma 1). The proof follows [AS24], and we provide it here for self-containment. For $0 \leq i \leq n$, consider the initial configuration C_i such that parties $p_1, p_2, \dots, p_i$ have input 1 and the rest have input 0. Let $\text{Val}(i)$ be the decision in the failure free execution that starts with configuration C_i (i.e., no Byzantine party at all). By validity, $\text{Val}(0) = 0$ and $\text{Val}(n) = 1$. Therefore, there must exist two adjacent configurations C_{i-1} and C_i ($1 \leq i \leq n$) such that

- $\text{Val}(i - 1) = 0$ and $\text{Val}(i) = 1$.
- The only difference between configurations C_{i-1} and C_i is the initial state of p_i .

Therefore, C_{i-1} and C_i are AS but FFD configurations. □

Lemma 2. *Let C and C' be two AS but FFD configurations at the beginning of round r in the BA protocol that tolerates one more Byzantine fault, then no honest party in executing the BA protocol can terminate before receiving any message in round $r + 1$.*

Proof (of Lemma 2). We prove via a contradiction. Assume that the BA protocol enables a party to terminate before receiving the $(r + 1)$ -round message if no faulty party is detected in round r . We show that there are two honest parties that finally output different values, hence conflicting with the consistency.

Let C and C' be two AS but FFD configurations at the beginning of round r ($r > 0$), and let p_i be the critical party of C and C' (i.e., the party holding different states in those two configurations). Let p_j, p_k be another two honest parties, and $M_{i \rightarrow j}$ and $M_{i \rightarrow k}$ be the messages p_i sends to p_j and p_k respectively during round r following the configuration C . Similarly, denote by $M'_{i \rightarrow j}$ and $M'_{i \rightarrow k}$ the messages that p_i sends to p_j and p_k respectively following the configuration C' .

Now define execution A (resp. execution A') as the failure-free execution from configuration C (resp. configuration C'). By our assumption, both p_j and p_k terminate and output the same value v (resp. v') before receiving any message at round $r + 1$ in execution A (resp. A'). Since C and C' are FFD, we have $v \neq v'$.

Now consider the hybrid execution A^* which is the same as execution A except that the adversary corrupts p_i , and sends $M'_{i \rightarrow k}$ instead of $M_{i \rightarrow k}$ to p_k during round r .¹² Since the BA protocol can tolerate one more Byzantine fault (namely p_i), the hybrid execution A^* is well-defined. We analyze p_j and p_k 's outputs as follows.

- Party p_j : From p_j 's view, it appears that p_i and all other parties are non-faulty in round r , which is the same as the view in execution A . Hence by our assumption p_j will terminate and output v at the end of round r or after sending the messages out in round $r + 1$.
- Party p_k : From p_k 's view, it appears that p_i and all other parties are non-faulty in round r , which is the same as the view in execution A' . Hence by our assumption p_k will terminate and output $v' \neq v$ at the end of round r or after sending the messages out in round $r + 1$.

¹² There is a subtle issue: p_i cannot send two distinct instructed messages without corresponding configurations (e.g., it cannot generate a signature of an honest party). However, we can still construct A^* since p_i is the critical party between C and C' , and this ensures (Byzantine) p_i can generate both $M_{i \rightarrow k}$ and $M'_{i \rightarrow k}$ which come from the same configuration except the part from p_i .

Therefore, honest parties p_j and p_k have different outputs in execution A^* , which conflicts with the consistency. This implies that both p_i and p_k needs to receive one more round message before termination. $\square$

Lemma 3 (Lemma 3 in [AS24]). *If C and C' are two AS but FFD configurations at the beginning of round r , then there are two AS but FFD configurations D and D' at the beginning of round $r + 1$ that each extends C, C' by one crash fault in round r .*

Note that the lemma considers only crash faults when extending configurations. This suffices for our purposes, since the Byzantine behavior beyond the crash fault is relevant only in the final round we analyze.

Proof (of Lemma 3). Let p_i be the only difference between C and C' , and w.l.o.g. assume that if p_i sends no message at all in round r then the failure free execution has a different output value from the failure free extension of C . Otherwise, it must have a different output value from the failure free extension of C' since C and C' are AS but FFD configurations. In that case, we use C' instead of C .

Now consider the $n + 1$ configurations $C_0, C_1, \dots, C_n = C$, where C_j denotes the configuration in which the adversary crashes p_i after it sends to p_j . Clearly any two consecutive configurations are AS because they differ in one party p_j that either received or did not receive p_i 's message.

Note that C_0 is the execution in which p_i crashes at the beginning of round r . From our above assumption, if we continue from C_0 without faults then the decision value is going to be different from the decision value in a failure-free extension of configuration C . Therefore, there must exist two consecutive configurations among $C_0, C_1, \dots, C_n = C$ that are AS but FFD. $\square$

Proof (of Theorem 2). Note that $\min\{f + 2, t + 1, n - 1\} = n - 1$ happens only when $t = n - 1$ and $f \geq n - 2$, where the $n - 1$ lower bound is trivial. Meanwhile, the $t + 1$ lower bound of the case $f = t$, i.e., $\min\{f + 2, t + 1\} = t + 1$, has been proven in [DRS90].

When $f < t < n - 1$, we start with Lemma 1, and apply Lemma 3 f times to reach two AS but FFD configurations at the beginning round $f + 1$. Then from Lemma 2 (by viewing r in Lemma 2 as $f + 1$), no honest party can complete (terminate or decide) the protocol before receiving messages in round $f + 2$. This concludes the lower bound of receive-round $\min\{f + 2, t + 1, n - 1\}$. $\square$

Remark 2. The lower bound in Theorem 2 also holds in the setup models assuming signatures, following the same proof strategy. Note that in such models, every message from p_i to p_j is (optionally) authenticated by p_i 's signature. This provides an additional way to detect faulty behaviors of p_i by collecting two conflict signed-messages of p_i . Nevertheless, this still does not affect the lower bound of $\min\{f + 2, t + 1, n - 1\}$ since corrupted parties can collude and generate conflict signed messages, as long as all parties appearing in the signature chain are faulty.¹³ In the above proof, we start with Lemma 1, and apply Lemma 3 f times to reach two “AS but FFD configurations” at the beginning round $f + 1$, where all crucial parties appearing in this process are Byzantine parties. Then the claimed bound is achieved by Lemma 2. The case of $f = t < n - 1$ trivially follows the lower bound $\min\{f + 2, t + 1\}$ as in [DRS90].

We discuss more on the subtle difference between the lower bound for crash faults [DRS90, AS24], and the lower bound for Byzantine faults in this paper. For crash faults, if some honest parties terminate in round $f + 1$, then BA cannot be achieved; therefore, they must send an additional message in round $f + 2$, leading to the $f + 2(s)$ lower bound. Nevertheless, they can safely determine their output value in round $f + 1$, since it will not change. In contrast, for Byzantine faults, if some honest parties determine their output value in round $f + 1$, then BA cannot be achieved. Since they have not yet terminated, they can still send an additional message in round $f + 2$. However, as shown above, their output values may be inconsistent in round $f + 1$, so they must determine their output value only after receiving the round- $f + 2$ messages, corresponding to the $f + 2(r)$ lower bound.

¹³ One can simply imagine that the signing secret keys are shared among faulty parties.

5.2 $f + 3(s)$ Lower Bound for Broadcast with Dishonest Majority

In this section, we prove that $f + 3(s)$ are necessary to achieve Byzantine broadcast against a corrupted majority when $f \leq 2t - n - 1$. Without loss of generality, we assume that in each round every party sends its entire view, together with a signature for authenticity. This assumption is justified for deterministic protocols for the following reasons: 1) any information sent by a party in a protocol is always captured by its view, and hence any protocol can be simulated under this assumption; and 2) any protocol that sends unsigned messages can be transformed into an equivalent protocol that includes signatures, preserving both correctness and security.

5.2.1 Warming-up: early stopping in round 2. To give intuition, we first show a simple example where a broadcast protocol fails by allowing termination in receive-round 2 when a party has not observed any Byzantine behavior.

Theorem 3. *Suppose $n > 4$ is odd, $t > n/2$, and $f = 0$, then any Byzantine broadcast protocol must have an execution with at least send-round 3.*

Proof. Let s be the designated sender and v_s is the sender's input value. Let P_1, P_2, P_3, P_4 be disjoint partitions of $\mathcal{P} \setminus \{s\}$ such that $|P_1| = |P_3|$ and $|P_2| = |P_4|$. Assume parties run a broadcast protocol that tolerates $t > n/2$ Byzantine parties and guarantees termination within (receive-)round 2 when $f = 0$. Consider the following five executions: $A_1^0, A_1^1, A_2^0, A_2^1, A_3$. We show, by contradiction, that honest parties in P_2 and P_3 output distinct bits in A_3 , thereby violating consistency.

- A_1^0 : In this execution, all parties are honest and $v_s = 0$. Let $M_{s \rightarrow i}^{0,r}$ and $M_{i \rightarrow j}^{0,r}$ denote the message sent from s to p_i and from p_i to p_j , respectively, in round r when $v_s = 0$.
Since there is no Byzantine party, if the protocol achieves $f + 2(r)$ early stopping, all parties must terminate with 0 in receive-round 2 for all $i \in [1..4]$.
- A_1^1 : In this execution, all parties are honest and $v_s = 1$. Similarly to A_1^0 , we use $M_{s \rightarrow i}^{1,r}$ and $M_{i \rightarrow j}^{1,r}$ to denote the message sent from s to p_i and from p_i to p_j , respectively, in round r when $v_s = 1$. By the assumption, all parties in P_i terminate with 1 for all $i \in [1..4]$.
- A_2^0 : In this execution, s and all parties in P_3 and P_4 are Byzantine. The sender s behaves honestly toward the parties in P_1 and P_2 as if $v_s = 0$ in rounds 1 and 2, i.e., s sends $M_{s \rightarrow i}^{0,1}$ and $M_{s \rightarrow i}^{0,2}$ to each $p_i \in P_1 \cup P_2$ in send-rounds 1 and 2, respectively. All parties in P_3 and P_4 behave honestly in round 1.
In round 2, each $p_j \in P_3 \cup P_4$ behaves as follows: it sends $M_{j \rightarrow k}^{0,2}$ to every $p_k \in P_1$, and $M_{j \rightarrow k'}^{1,2}$ to every $p_{k'} \in P_2$ (this is feasible since s is also a Byzantine). Since the views of $p_k \in P_1$ in A_1^0 and A_2^0 are identical by the end of receive-round 2, all parties in P_1 terminate with 0 in receive-round 2.
From round 3 onward, s and all parties in P_4 send no messages, while each party $p_j \in P_3$ behaves honestly as if it had received $M_{s \rightarrow j}^{1,1}$ in round 1 and $M_{j' \rightarrow j}^{1,2}$ from $p_{j'} \in P_4$ in round 2.
Since the parties in P_2 detect Byzantine behaviors of s and P_4 , they may continue for additional rounds. However, to satisfy the consistency property of BA, all parties in P_2 must eventually terminate with 0.
- A_2^1 : This execution is symmetric to A_2^0 . In this execution, s and all the parties in P_1 and P_2 are Byzantine. The sender s behaves honestly toward the parties in P_3 and P_4 as if $v_s = 1$ in rounds 1 and 2, i.e., s sends $M_{s \rightarrow i}^{1,1}$ and $M_{s \rightarrow i}^{1,2}$ to each $p_i \in P_3 \cup P_4$ in send-rounds 1 and 2, respectively. All parties in P_1 and P_2 behave honestly in round 1.
In round 2, each $p_j \in P_1 \cup P_2$ behaves as follows: it sends $M_{j \rightarrow k}^{0,2}$ to every $p_k \in P_3$ and $M_{j \rightarrow k'}^{1,2}$ to every $p_{k'} \in P_4$. From round 3 onward, s and all parties in P_1 send no messages, while each party $p_j \in P_2$ behaves honestly as if it had received $M_{s \rightarrow j}^{0,1}$ in round 1 and $M_{j' \rightarrow j}^{0,2}$ from $p_{j'} \in P_1$ in round 2.
By analogous reasoning, all parties in P_4 terminate with 1 in receive-round 2, and all parties in P_3 eventually terminate with 1 due to consistency.

- A_3 : In this execution, s and all parties in P_1 and P_4 are Byzantine. The sender s sends $M_{s \rightarrow i}^{0,1}$ and $M_{s \rightarrow i}^{0,2}$ to every $p_i \in P_2$, and sends $M_{s \rightarrow i'}^{1,1}$ and $M_{s \rightarrow i'}^{1,2}$ to every $p_{i'} \in P_3$ in send-rounds 1 and 2, respectively. All parties in P_1 and P_4 behave honestly in round 1.

In round 2, each $p_j \in P_1$ sends $M_{j \rightarrow k}^{0,2}$ to every $p_k \in P_2 \cup P_3$, while each $p_{j'} \in P_4$ sends $M_{j' \rightarrow k}^{1,2}$ to every $p_k \in P_2 \cup P_3$. From round 3 onward, s and the parties in $P_1 \cup P_4$ send no messages. Since the parties in $P_2 \cup P_3$ detect Byzantine parties, they may continue for additional rounds.

We now analyze the outputs of honest parties in P_2 and P_3 . If some $p_i \in P_2$ outputs 1, then this contradicts to A_2^0 , since p_i cannot distinguish views between A_2^0 and A_3 . Therefore, $p_i \in P_2$ will eventually output 0. Similarly, due to the indistinguishable views between A_2^1 and A_3 , a party $p_i \in P_3$ will eventually output 1. Consequently, parties in P_2 and P_3 output different values in this execution, leading to a contradiction.

As a result, Byzantine broadcast cannot guarantee termination in receive-round 2 without observable Byzantine behavior. This completes the proof. $\square$

5.2.2 Generalization. In general, we have the following theorem.

Theorem 4. *Suppose $n > 4$, $t < n - 1$ and $0 \leq f \leq 2t - n - 1$, then any Byzantine broadcast protocol must have an execution with at least send-round $f + 3$.*

Proof. The theorem trivially holds for $t \leq n/2$, since this implies $f < 0$. Thus, it suffices to consider the case $n/2 < t < n$. Let $r = 2t - n$, so that $0 \leq f < r < t$. By Lemmas 1 and 3, there exist two AS but FFD configurations, C and C' , at the beginning of send-round r . Let v and v' denote the decision values in the failure-free executions of C and C' , respectively. Note that these configurations contain $r - 1$ crashed (i.e., Byzantine) parties.

Let party p_{i^*} be the only party on which C and C' differ. Excluding the crashed parties and p_{i^*} , the remaining $n - r = 2n - 2t \geq 4$ parties can be partitioned into four disjoint groups P_1 , P_2 , P_3 , and P_4 such that $|P_1| = |P_3|$ and $|P_2| = |P_4|$. Note that the only difference between C and C' is whether p_{i^*} received a message from the previous critical party $p_{i'}$ in the previous round.

- A_1^0 : In this execution, p_{i^*} and all parties in $P_1 \cup P_2 \cup P_3 \cup P_4$ are honest following the configuration C . Let $M_{i \rightarrow j}^{C,r}$ denote the message sent from p_i to p_j in round r and round r in the honest extension of C (including the case $i = i^*$). Similarly, let $M_{j \rightarrow k}^{C,r+1}$ denote the message sent from p_j to p_k in round $r + 1$ in the honest extension of C . Note that there are only $f = r - 1$ Byzantine parties in this execution. Consequently, if the protocol achieves $f + 2(r)$ early stopping, then all parties in P_i terminate with v in receive-round $r + 1$ for all $i \in [1..4]$.

- A_1^1 : In this execution, p_{i^*} and all parties in $P_1 \cup P_2 \cup P_3 \cup P_4$ are honest following the configuration C' . Similarly to the execution A_1^0 , we use $M_{i \rightarrow j}^{C',r}$ to denote the message sent from p_i to p_j in round r in the honest extension of configuration C' (including the case $i = i^*$), and $M_{j \rightarrow k}^{C',r+1}$ to denote the message sent from p_j to p_k in round $r + 1$. By the assumption, all parties in P_i terminate with v' for all $i \in [1..4]$.

- A_2^0 : In this execution, p_{i^*} and all parties in P_3 and P_4 are Byzantine. The party p_{i^*} behaves honestly toward the parties in P_1 and P_2 as if it starts from configuration C in rounds r and $r + 1$. Accordingly, p_{i^*} sends $M_{i^* \rightarrow j}^{C,r}$ and $M_{i^* \rightarrow j}^{C,r+1}$ to each $p_j \in P_1 \cup P_2$ in send-rounds r and $r + 1$, respectively. All parties in P_3 and P_4 behave honestly in round r .

In round $r + 1$, each $p_j \in P_3 \cup P_4$ behaves as follows: it sends $M_{j \rightarrow k}^{C,r+1}$ to every $p_k \in P_1$, and $M_{j \rightarrow k'}^{C',r+1}$ to every $p_{k'} \in P_2$. Since the views of $p_j \in P_1$ in A_1^1 and A_2^0 are identical by the end of receive-round $r + 1$, all parties in P_1 terminate with v in receive-round $r + 1$.

From round $r + 2$ onward, p_{i^*} and all parties in P_4 send no messages, while each party $p_j \in P_3$ behaves honestly as if it had received $M_{i^* \rightarrow j}^{C',r}$ in round r and $M_{j' \rightarrow j}^{C',r+1}$ from $p_{j'} \in P_4$ in round $r + 1$.

Since the parties in P_2 detect Byzantine behaviors of p_{i^*} and the parties in P_4 , they may continue for additional rounds. However, to satisfy the consistency property of BA, all parties in P_2 must eventually terminate with v .

- A_2^1 : This execution is symmetric to A_2^0 . In this execution, p_{i^*} and all the parties in P_1 and P_2 are Byzantine. The party p_{i^*} behaves honestly toward the parties in P_3 and P_4 as if it starts from configuration C' in rounds r and $r + 1$. Accordingly, p_{i^*} sends $M_{i^* \rightarrow j}^{C',r}$ and $M_{i^* \rightarrow j}^{C',r+1}$ to each $p_j \in P_3 \cup P_4$ in send-rounds r and $r + 1$, respectively. All parties in P_1 and P_2 behave honestly in round r .

In round $r + 1$, each $p_j \in P_1 \cup P_2$ behaves as follows: it sends $M_{j \rightarrow k}^{C',r+1}$ to every $p_k \in P_3$ and $M_{j \rightarrow k'}^{C',r+1}$ to every $p_{k'} \in P_4$. From round $r + 2$ onward, p_{i^*} and the parties in P_1 send no messages, while each party $p_j \in P_2$ behaves honestly as if it had received $M_{i^* \rightarrow j}^{C',r}$ in round r and $M_{j' \rightarrow j}^{C',r+1}$ from $p_{j'} \in P_1$ in round $r + 1$.

By analogous reasoning, all parties in P_4 terminate with v' in receive-round $r + 1$, and all parties in P_3 must eventually terminate with v' due to consistency.

- A_3 : In this execution, p_{i^*} and all parties in P_1 and P_4 are Byzantine. The party p_{i^*} sends $M_{i^* \rightarrow i}^{C',r}$ and $M_{i^* \rightarrow i}^{C',r+1}$ to every $p_i \in P_2$, and sends $M_{i \rightarrow k'}^{C',r}$ and $M_{i \rightarrow k'}^{C',r+1}$ to every $p_{i'} \in P_3$ in send-rounds r and $r + 1$, respectively. All parties in P_1 and P_4 behave honestly in round r .

In round $r + 1$, each $p_j \in P_1$ sends $M_{j \rightarrow k}^{C',r+1}$ to every $p_k \in P_2 \cup P_3$, while each $p_{j'} \in P_4$ sends $M_{j' \rightarrow k}^{C',r+1}$ to every $p_k \in P_2 \cup P_3$. From round $r + 2$ onward, p_{i^*} and the parties in $P_1 \cup P_4$ send no messages. Since the parties in $P_2 \cup P_3$ detect Byzantine parties, they may continue for additional rounds.

We now analyze the outputs of honest parties in P_2 and P_3 . If some $p_i \in P_2$ outputs v' , then it contradicts to A_2^0 , since p_i cannot distinguish views between A_2^0 and A_3 . Therefore, $p_i \in P_2$ will eventually output v . Similarly, due to the indistinguishable views between A_2^1 and A_3 , a party $p_i \in P_3$ will eventually output v' . Consequently, parties in P_2 and P_3 output different values in this execution, leading to a contradiction.

As a result, if $n > 4$, $t < n - 1$ and $0 \leq f \leq 2t - n - 1$, then every BA protocol must have an execution with at least $f + 3(s)$. $\square$

6 An Optimal Broadcast Protocol Against Omission Faults

In this section, we introduce a (refined) round-optimal early-stopping Broadcast protocol that can tolerate omission faults in Fig. 4. Note that our protocol can be turned into a round-optimal Consensus protocol simply by running the Broadcast protocol in parallel for each party as the sender. Therefore, the protocol proves the tightness of our refined round complexity result on early-stopping BA with omission faults (see Section 2 for the discussion).

Let s be the designated sender and let $v_s \in \mathcal{D}$ be its input value. m_p^r denotes the message sent from p in round r , and $\mathcal{P}_p^r$ denotes the set of parties that appear to be honest in round r from p 's perspective. A party that sends $v \in \mathcal{D}$ is not included in $\mathcal{P}_p^r$ since in this case p terminates in the next send-round, rendering $\mathcal{P}_p^r$ meaningless. b_{halt} is a boolean flag indicating halt.

Theorem 5. *The protocol in Fig. 4 achieves Byzantine Broadcast against omission faults, and the protocol runs at most $\min\{f + 2(s), t + 1(r)\}$.*

Proof. To prove that the protocol achieves Byzantine Broadcast, we need to show that the protocol satisfies the following four properties: validity, consistency, termination, and early stopping. We consider $t > 0$ case only since otherwise it is trivial.

Validity. If the sender is honest, then it sends $(s, v_s, 1)$ in send-round 1. All honest parties receive $(s, v_s, 1)$ in receive-round 1, and by Line 13, they set $b_{\text{halt}} \leftarrow \text{true}$ and $v_{\text{out}} \leftarrow v_s$. Consequently, all honest parties proceed to send-round 2 and terminate with output v_s by executing Line 8.

The sender s has an input $v_s \in \mathcal{D}$. Each $p \in \mathcal{P}$ in each round $1 \leq r \leq t+1$ does the following:

```

1 : Initialization : // initialize the set  $P_p^0$ , the halt flag  $b_{\text{halt}}$ , and the output/send value  $v_{\text{out}}$ 
2 :  $P_p^0 \leftarrow \mathcal{P}$ ;  $b_{\text{halt}} \leftarrow \text{false}$ ;  $v_{\text{out}} \leftarrow \text{Null}$  // the initial value of parties except for the sender is Null
3 : if  $p = s$  :  $b_{\text{halt}} \leftarrow \text{true}$ ;  $v_{\text{out}} \leftarrow v_s$  // the initial value of the sender is  $v_s$ 
4 :  $m_p^1 \leftarrow (p, v_{\text{out}}, 1)$ 
5 : for  $r = 1$  to  $t+1$  :
6 :   // send-round  $r$ 
7 :   Send  $m_p^r$  to  $q \in \mathcal{P}$ 
8 :   if  $b_{\text{halt}}$  : return  $v_{\text{out}}$  // terminate in send-round  $r$ 
9 :   // receive-round  $r$ 
10 :  Receive  $m_q^r$  from each  $q \in \mathcal{P}$ 
11 :   $P_p^r \leftarrow P_p^{r-1} \cap \{q \in \mathcal{P} : m_q^r = (q, \text{Null}, r) \vee \exists r' \leq r \text{ such that } m_q^{r'} = (q, \perp, r')\}$ 
12 :  if  $\exists (q, v) \in \mathcal{P} \times \mathcal{D}$  s.t.  $m_q^r = (q, v, r)$  : // in other words,  $v \notin \{\perp, \text{Null}\}$ 
13 :     $b_{\text{halt}} \leftarrow \text{true}$ ;  $v_{\text{out}} \leftarrow v$ ;  $m_p^{r+1} \leftarrow (p, v_{\text{out}}, r+1)$ 
14 :  elseif  $|P_p^r| > n - r$  :
15 :     $b_{\text{halt}} \leftarrow \text{true}$ ;  $v_{\text{out}} \leftarrow \perp$ ;  $m_p^{r+1} \leftarrow (p, v_{\text{out}}, r+1)$ 
16 :  else :
17 :     $v_{\text{out}} \leftarrow \text{Null}$ ;  $m_p^{r+1} \leftarrow (p, v_{\text{out}}, r+1)$ 
18 :  if  $r = t+1$  : return  $v_{\text{out}}$  // the rule for receive-round  $t+1$ 

```

Fig. 4. $\min\{f + 2(s), t + 1(r)\}$ early-stopping Byzantine Broadcast protocol tolerating omission faults. The input domain $\mathcal{D}$ does not contain special symbols $\{\text{Null}, \perp\}$.

Termination. In receive-round $t + 1$, all honest parties that have not yet terminated execute Line 18 and terminate.

Consistency. Let r be the smallest round number such that some party terminates in either send- or receive-round r , and no honest party terminates in any round $r' < r$. Let p be a party that terminates in round r with output v . If $r = 1$, then $p = s$ is the only possibility, and consistency holds by validity.

Now assume $2 \leq r < t + 1$. We argue that there cannot exist an honest party $q \neq p$ such that terminates in round r with output $w \neq v$. Suppose, for contradiction, that such a party q exists. The only possibility is that $\perp \in \{v, w\}$, since q can terminate in round $r < t + 1$ by only executing Line 8, and $b_{\text{halt}} \leftarrow \text{true}$ is set only by executing Line 13 or 15.

First, assume $w = \perp$ and $v \neq \perp$. This implies that p received $(\cdot, v, r - 1)$ in round $r - 1$, while q did not. Since q terminates in round r , it must hold that $|P_q^{r-1}| > n - r + 1$ in round $r - 1$. On the other hand, for p to receive $(x, v, r - 1)$ in receive-round $r - 1$, there must exist another party $x \notin \{p, q\}$ such that x has received $(y, v, r - 2)$ for some $y \notin \{p, q, x\}$. Continuing inductively, this requires at least $r - 1$ faulty parties to sustain this v -message while preventing p from receiving $(\cdot, v, \cdot)$ before round $r - 1$. Moreover, these faulty parties must have not sent v -messages to q , since otherwise q would also terminate with v . Thus, q can detect at least $r - 1$ faulty parties that exhibit omission faults (i.e., failing to send messages in some round $r' < r$) without sending $(\cdot, \perp, r'')$ for some $r'' < r'$. Consequently, we obtain $|P_q^{r-1}| \leq n - r + 1$ as a contradiction. Therefore, if p is among the first group of parties to terminate and does so with output $v \in \mathcal{D}$ in round $r < t + 1$, then every other party that terminates in the same round must also output v . The remaining argument is straightforward: once p terminates with v in round r , any party that has not yet terminated will terminate in send-round $r + 1$ with output v upon receiving the message (p, v, r) .

On the other hand, suppose that p terminates with $\perp$. This implies that no v -message exists in round $r - 1$, since at most $r - 2$ parties can send v by round $r - 1$ (as $|P_p^{r-1}| > n - r + 1$). Therefore, no honest party can terminate with v . Furthermore, by send-round $t + 1$, all honest parties must terminate with either $\perp$ or v . Thus, if an honest party q terminates by send-round $t + 1$, then it must output $\perp$. If q terminates in receive-round $t + 1$, then it must hold that $|P_q^{t+1}| > n - t - 1$, since there are at most t faulty parties. Consequently, q executes Line 15 and terminate with $\perp$.

Now assume that $r = t + 1$. As established above, if any honest party terminates by round t , then the consistency property is satisfied. Suppose instead that no honest party terminates by round t . If a v -message exists by round t , then $t - 1$ faulty parties must have terminated by that round, since they must have sent $(\cdot, v, \cdot)$ in the preceding rounds and executed Line 8. Let x be the final faulty party. If x sends (x, v, t) to an honest party p , then p will send $(p, v, t + 1)$ to all honest parties in send-round $t + 1$. Consequently, all honest party terminate with v by receive-round $t + 1$. Otherwise, x does not send (x, v, t) to any honest party. In this case, no honest party receives $(\cdot, v, t + 1)$ in round $t + 1$. Since there are at most t faulty parties, it follows that $|P^{t+1}| > n - t - 1$. Consequently, all honest parties terminate consistently with $\perp$.

This completes the proof for consistency.

Early Stopping. If no honest party has sent $(\cdot, v, \cdot)$ by round $f + 1$, then for any honest party p that has not terminated by receive-round $f + 1$, the set P_p^{f+1} contains all honest parties, since they must send either $(\cdot, \text{Null}, f + 1)$ or $(\cdot, \perp, \cdot)$. Therefore, any such party p executing Line 14 observes that $|P_p^{f+1}| \geq n - f > n - f - 1$ since there are at most f faulty parties, and thus p terminates with $\perp$ in send-round $f + 2$.

Otherwise, there exists an honest party that has sent $(\cdot, v, \cdot)$ by round $f + 1$, and hence all honest parties terminate with v by send-round $f + 2$. $\square$

7 Revisiting Cloture Votes

In this section we review the Cloture Votes protocol by Berman and Garay [BG93], and present a revised version that achieves $\min\{f + 2(r), t + 1(r)\}$ and tolerates $t < n/4$ Byzantine corruptions.

7.1 Review of the Cloture Votes Protocol

Cloture is a parliamentary procedure (aka “parliamentary guillotine”) to curtail unnecessary long debates. In the Cloture Votes protocol [BG93], the unanimous will of the honest parties (akin to a parliamentary supermajority) may curtail the debate if they are required to send a big message over the pre-set threshold. This is facilitated by having the parties initiate a new so-called Debate protocol (see Fig. 5) in each round, which either ends quickly, with the conclusion “continue” or “terminate with the default value”, or lasts through many rounds. By repeatably invoking Debate, the Cloture Votes protocol solves the Consensus problem with message complexity $O(n^5)$ in the setting $n > 4t$.

The Debate Protocol. The Debate protocol (Fig. 5) extends the basic EIG protocol by additionally introducing fault masking and early stopping. Namely, if an honest party p detects q as Byzantine, then all $(\cdot, q, \cdot)$ entries in $\mathcal{IN}_{q \rightarrow p}^r$ are replaced with the default value v_0 , i.e., $(\cdot, q, v_0)$. Moreover, if in round r the value of some node σ has been fixed (or predicted) by all honest parties, then all σ -projections are unnecessary to send, making these branches stop early. Consequently, the Debate protocol has much shorter messages compared with the original EIG protocol.

We defined reliable node σp as in Definition 3. Note that in EIG trees, there are also nodes where σ includes some honest parties. Nevertheless, [BG93] protocol guarantees that all honest parties resolve the consistent value of a reliable node σp in round $|\sigma p| + 1$. Once the consistent value of an ancestor node was resolved, the values of its descendant nodes become irrelevant, as they do not affect the final output of the protocol. Therefore, we only need to consider reliable nodes and unreliable nodes.

As mentioned above, the consistent value of a reliable node σq must be resolved in round $|\sigma q| + 1$ in the protocol. Consequently, if a node σq is not resolved by that round, then q must be a Byzantine party.

Definition 9 ((Fully/Partially) Corrupted Nodes in [BG93]). *A node σq that has not been resolved by round $|\sigma q| + 1$ (by some party p) is called a corrupted node (with respect to p).*

Moreover, at the end of round $|\sigma q| + 1$, if no honest party resolves σq , then it is called a fully corrupted node; if some honest parties resolve σq while others do not, then it is called a partially corrupted node.

Furthermore, if σq is a fully corrupted node, we say that q *fully corrupts* σq . If σq is a partially corrupted node, we say that q *partially corrupts* the node σq .

The protocol is designed such that if an honest party p resolves a node σq in round r , then all other honest parties will consistently resolve σq in round $r + 1$, i.e., in the next round. This implies that, to prevent the protocol from terminating in the next round, at least one of Byzantine parties must fully corrupt a node in each round. Otherwise, all honest parties will resolve all unresolved nodes at the current level, causing the protocol to terminate. However, if a Byzantine party q fully corrupts a node in round r , then all honest parties will detect q as a Byzantine party in round $r + 1$, since any reliable node must be resolved by that round. Once a Byzantine party q is globally detected, the protocol ensures that q can no longer corrupt a nodes by fault masking, effectively excluding q from the agreement process. Therefore, Byzantine parties can continue to disrupt the agreement process only until round $\min\{f + 2, t + 1\}$, as at least one Byzantine party must be excluded in each round.

When the Debate protocol is invoked in the Cloture Votes protocol in round r , there are at least $r - 1$ Byzantine parties that have been detected and hence the messages from them are replaced with the default value. This guarantees that in the worst case, an invocation of the Debate subprotocol in round r will last for at most $t - r + 2$ rounds,¹⁴ and hence terminate in round $t + 1$.

The Debate protocol itself suffers from exponential message complexity at the adversary’s will. If Byzantine parties continue to disrupt the agreement process until round $\min\{f + 2, t + 1\}$, then at most one fully corrupted node can exist in each round, resulting in a message complexity of $O(n^2)$. However, suppose that j Byzantine parties fully corrupt nodes as many as possible in each round. In this scenario, the Byzantine parties can sustain this behavior up to $\lfloor f/j \rfloor$ rounds, resulting in a total of $j^{\lfloor f/j \rfloor}$ fully corrupted nodes. This expression is maximized when $j = e$, but since j must be the natural number, the maximum occurs at $j = 3$. Consequently, the worst-case message complexity of the protocol becomes $O(3^{f/3}) = O(3^{n/12})$.

¹⁴ This differs from [BG93] where the protocol is initialized at round $r = 0$ rather than at round $r = 1$ here.

```

Debatepk(v) run by party p:
1 : // k denotes the current round when Debate is invoked in the Cloture Votes
2 : // v denotes the initial value
3 : Initialization : // create root nodes ε for both  $\mathcal{IT}_p^k$  and  $\mathcal{RT}_p^k$ 
4 :  $\mathcal{IT}_p^k(\varepsilon) \leftarrow v$ ;  $\mathcal{RT}_p^k(\varepsilon) \leftarrow \perp$ ;  $M_p^k \leftarrow U_p^k \leftarrow \{\varepsilon\}$ 
5 : //  $M_p^k$ : set of nodes whose values need to be shared
6 : //  $U_p^k$ : set of nodes whose values are undecided and child nodes are required
7 : for  $r = k$  to  $t + 1$  : // the protocol runs  $t - k + 2$  rounds
8 :    $\mathcal{OUT}_p^r \leftarrow \bigcup_{\sigma \in M_p^k} \{(\sigma, p, \mathcal{IT}_p^k(\sigma))\}$ 
9 :   Send  $\mathcal{OUT}_p^r$  to  $q \in \mathcal{P}$ 
10 :   $M_p^k \leftarrow \bigcup_{\sigma \in U_p^k, q \in \mathcal{P} \setminus [\sigma]} \{\sigma q\}$ 
11 :  Receive  $\mathcal{IN}_{q \rightarrow p}^r$  from each party  $q \in \mathcal{P}$ 
12 :  for every  $\sigma q \in M_p^k$ 
13 :    if  $\exists(\sigma, q, v) \in \mathcal{IN}_{q \rightarrow p}^r$  :  $\mathcal{IT}_p^k(\sigma q) \leftarrow v$ 
14 :    else :  $\mathcal{IT}_p^k(\sigma q) \leftarrow 0$ 
15 :  if  $r = k$  and received “I am done” from  $q$  :  $\mathcal{IT}_p^k(q) \leftarrow 1$ 
16 :  // fault masking
17 :  for  $\sigma q \in M_p^k$  such that  $q \in \mathcal{F}_p$  :  $\mathcal{IT}_p^k(\sigma q) \leftarrow 0$ 
18 :  apply Resolve Rules
19 :  // early stopping
20 :  for  $\sigma$  such that  $\mathcal{RT}_p^k(\sigma) \neq \perp$  :
21 :    remove all descendants of  $\sigma$  from  $U_p^k$ 
22 :   $U_p^k \leftarrow \bigcup_{\sigma \in U_p^k, q \in \mathcal{P} \setminus [\sigma]} \{\sigma q\}$ 
23 :  if  $\mathcal{RT}_p^k(\varepsilon) \neq \perp$  :
24 :    for every  $\sigma \in M_p^k$  :
25 :       $\mathcal{OUT}_p^{r+1} \leftarrow \bigcup_{\sigma \in M_p^k} \{(\sigma, p, \mathcal{IT}_p^k(\sigma))\}$ 
26 :      Send  $\mathcal{OUT}_p^{r+1}$ 
27 :      // the above three lines are not explicitly stated in [BG93] but essential
28 :  return  $\mathcal{RT}_p^k(\varepsilon)$ 

```

Fig. 5. The Debate protocol introduced by [BG93].

ClotureVotes(v) run by party p with initial value v_p:	
1 :	Initialization :
2 :	<i>// initialize the threshold S of the message size, the halt flag b_{done}, and the output value v_{out}</i>
3 :	$S \leftarrow 2n^3$; $b_{\text{done}} \leftarrow \text{false}$; $v_{\text{out}} \leftarrow \perp$
4 :	if round $r = 1$: start $\text{Debate}_p^1(v_p)$
5 :	if round $2 \leq r \leq t + 1$:
6 :	if b_{done} :
7 :	Send “I am done” and return v_{out}
8 :	run subprotocol Debate_p^k for $k < r$
9 :	if $\exists \ell \leq r$ such that $ M \geq S$:
10 :	<i>// M is the set of nodes whose values are needed to share with others</i>
11 :	start $\text{Debate}_p^r(0)$ <i>// vote for Cloture</i>
12 :	else : start $\text{Debate}_p^r(1)$
13 :	apply Resolve Rules
14 :	if $\exists k < r$ such that $\mathcal{RT}_p^k(\varepsilon) = 0$ in Debate_p^k :
15 :	return 0
16 :	elseif $ \mathcal{F}_p + 1 < r$:
17 :	$b_{\text{done}} \leftarrow \text{true}$; $v_{\text{out}} \leftarrow \min_{1 \leq k \leq r} \mathcal{RT}_p^k(\varepsilon)$
18 :	if $r = t + 1$: return $\min_{1 \leq k \leq r} \mathcal{RT}_p^k(\varepsilon)$

Fig. 6. The Cloture Votes protocol introduced by [BG93]. The early stopping condition in [BG93] is set as $(|\mathcal{F}_p| < r)$ (after accounting for the starting round $r = 0$); we think it is a flaw and present it as $(|\mathcal{F}_p| + 1 < r)$ here. Note also that [BG93] only briefly described the early-stopping variant without presenting full pseudocode; here we reconstruct it from the natural-language description.

The Clotures Votes Protocol. In the Cloture Votes protocol (Fig. 6), each party concurrently runs several Debate protocols Debate^k , each started in round k . Protocol Debate^k maintains its own $\mathcal{IT}$ and $\mathcal{RT}$ trees, which we denote by $\mathcal{IT}^k$ and $\mathcal{RT}^k$, respectively. Thus, in round $r > k$, the values of nodes σ with $|\sigma| = r - k + 1$ are recorded in $\mathcal{IT}^k(\sigma)$. Accordingly, we modify the definition of a corrupted node: a node σq in Debate^k that has not been resolved by round $|\sigma q| - k + 2$ (by some party p) is called a corrupted node (with respect to p). Note that the set of Byzantine parties $\mathcal{F}_p$ detected by party p is shared among all Debate protocols $\{\text{Debate}_p^k\}_{1 \leq k \leq r}$.

While running Debate^1 as the basic agreement process, in round r with $1 < r \leq t + 1$ each party starts a new Debate protocol $\text{Debate}^r(v)$ with initial value v . Here $v = 0$, triggered by a super big message needs to be sent in this round, means the party votes for cloture and 1 otherwise—this is exactly what a parliamentary cloture vote is about: although the cloture vote itself may be subject to debate, once the “filibuster speeches” exceed a certain polynomial length, the cloture vote becomes unanimous and the deliberation terminates. At the same time, the early stopping property of the Debate protocol guarantees that an invocation in round r involves at least $r - 1$ nonbinding Byzantine parties (i.e., parties that have been detected), and thus terminates within $t - r + 2$ rounds.

However, this protocol is in fact a $\min\{f + 3(s), t + 1(r)\}$ Consensus protocol, due to the following two reasons.

First, the protocol requires parties to send “I am done” message after determining their decision value. As shown in the previous section, honest parties may reach a decision in round $f + 2$ in the worst case. Consequently, they must send “I am done” message in round $f + 3$, although no honest party needs to receive it.

Second, the consistency of the Debate protocol relies on *Liveness*: if an honest party p resolves a node σ in round r , then all other honest parties must also resolve σ consistently by round $r + 1$. To achieve this property, p must send a message in round $r + 1$, though this requirement is not explicitly stated in [BG93]. Again, in the worst case, honest parties may decide their output value in some Debate^k in round $f + 2$, which forces them to send the corresponding messages in round $f + 3$ even though no honest party needs to receive them.

7.2 Resolve Rules and Fault Detection

Note that the following resolve rules are common for both [BG93] and our revised version.

A party p applies the following *resolve rules* in round r :

1. IT resolve rule
In $\mathcal{IT}_p^k$, for all σ such that $|\sigma| = r - k$, if there are more than $n/2 + t - |\sigma|$ child nodes σx such that $\mathcal{IT}_p^k(\sigma x) = v$, then $\mathcal{RT}_p^k(\sigma) := v$.
2. RT resolve rule
In $\mathcal{RT}_p^k$, for any node σ with $|\sigma| \leq r - k$, if a majority of child nodes σx satisfy $\mathcal{RT}_p^k(\sigma x) = v$, then $\mathcal{RT}_p^k(\sigma) := v$.
3. Round $t + 1$ rule
In round $t + 1$, $\mathcal{RT}_p^k(\sigma) := \mathcal{IT}_p^k(\sigma)$ such that $|\sigma| = t + 2 - k$.

Remark 3. Note that [BG93] defined “ σ is v -common” and proposed three criteria of that, which are exactly the above resolve rules. In this work we use two trees, $\mathcal{IT}$ and $\mathcal{RT}$ to precisely describe the procedure of protocols (including ours). A party p resolves a node σ by setting $\mathcal{RT}_p(\sigma) = v$ if and only if p is confirmed that all honest parties will resolves σ in their resolution trees, which is equivalent to the definition “ σ is v -common” in [BG93].

A party p applies the following *fault detection rule*:

1. Fault detection rule in round r in [BG93]
In $\mathcal{IT}_p^k$, for any node σx with $|\sigma x| = r - k$, if there are more than t distinct $y \in \mathcal{P}$ such that $\mathcal{IT}_p^k(\sigma xy) \neq \mathcal{IT}_p^k(\sigma x)$ or $y \in \mathcal{F}_p$, then add x to $\mathcal{F}_p$.

7.3 Revised Cloture Votes: An $f + 2(r)$ Consensus Protocol

We now present the revised Cloture Votes protocol. We start by observing several points of the original description that merit refinement. In [BG93], the protocol was designed to run $t + 1$ rounds and was later adapted to support $f + 2$ early stopping. However, this adaption is incomplete and introduces inconsistencies in the description. For example, in Debate_p^k , party p must send a message once after determining the output of Debate_p^k to ensure consistency under early stopping. Since the original protocol was structured to continue round $t + 1$, this requirement was omitted. Moreover, [BG93] provides only a brief outline of the early-stopping variant without full pseudocode, leaving some operational details to be inferred.

While the protocol in [BG93] is secure, the proof as presented misses some steps in detail. Specifically, Lemma 3.1 in [BG93] only addresses the case where $|F_p| + 1 < r$, leaving open the possibility that another honest party q could terminate inconsistently after round r . In our revised version, we provide a proof that covers all cases and demonstrate that the revised protocol is a Byzantine Consensus protocol guaranteeing termination by $\min\{f + 2(r), t + 1(r)\}$.

In our revision, we first switch the meaning of 1 and 0 for ‘Cloture’ in the invocation of the Debate protocol. In the Cloture Votes protocol in [BG93], the “I am done” message from p is indeed unnecessary, as it merely indicates that p will vote ‘Continuation’ permanently. If silence is simply interpreted as ‘Continuation’ at the beginning of each Debate, then the “I am done” message becomes redundant. Therefore, in our revised Cloture Votes protocol (Fig. 8), we redefine $\text{Debate}(0)$ to represent ‘Continuation’ and $\text{Debate}(1)$ to represent ‘Cloture’, since the default value for silence in Debate is 0.

We then introduce a new rule for handling silence in the Debate protocol: if there is no $(\sigma, q, \cdot)$ in $\mathcal{IN}_{q \rightarrow p}^r$, then the value of σq (in $\mathcal{IT}_p$) is assumed to be the same as that of σ , i.e., $\mathcal{IT}_p(\sigma q) := \mathcal{IT}_p(\sigma)$. This rule effectively ensures Liveness although there is no σ -projection from honest parties that have already resolved the node σ in the previous round. Naturally, no party can resolve a node before Debate starts. Therefore, if the first message in Debate is missed, then it is replaced by 0, consistent with the rule above. With this modification, honest parties no longer need to send σ -projections when σ is resolved in the current round. Since they only send projections about unresolved nodes, the message size is reduced by a factor of n . Consequently, the threshold for ‘Cloture’ becomes $2n^2$, and the message complexity is reduced to $O(n^4)$.

We adopt the following (simplified) *fault detection rule* in the revised protocol. The original rule also remains valid, since it fully subsumes the new one.

1. Fault detection rule (in our revised protocols)

In $\mathcal{IT}_p^k$, for any node σx with $|\sigma x| = r - k$, if $\mathcal{RT}_p^k(\sigma') = \perp$ for all $\sigma' \subseteq \sigma$ and σx has at most $n/2 + t - |\sigma x|$ distinct child nodes σxy such that $\mathcal{IT}_p^k(\sigma xy) = v$ for some $v \in \mathcal{D}$, then add x to $\mathcal{F}$.

Informally speaking, if a node σx recorded in $\mathcal{IT}$ in round $r - 1$ is not resolved by the IT resolve rule, then x is added to $\mathcal{F}$. Note that the fault detection rule is applied prior to the application of the resolve rules. Therefore, the term *resolve* is not used in the condition of the fault detection rule.

The revised Debate protocol and Cloture Votes protocol are shown in Fig. 7 and Fig. 8, respectively, where the resolves rules are identical in both the original protocols and the revised ones.

Theorem 6. *The revised Cloture Votes protocol in Fig. 8 is a Byzantine Consensus protocol that guarantees termination by $\min\{f + 2(r), t + 1(r)\}$ with the message complexity of $O(n^4)$, provided that $n > 4t$.*

We provide the following lemmas to prove Theorem 6. Note that the meaning of 0 and 1 in Debate has been switched here.

Lemma 4. *If all honest parties start with the same input value v in Debate^r , then all honest parties output v in Debate^r in round r .*

Proof. In round r , since all honest parties send $(\varepsilon, \cdot, v)$, there exist distinct parties $x_1, \dots, x_{n-t}$ such that $\mathcal{IT}^r(x_i) = v$ for all $i \in [1..n-t]$. Since $n > 4t$, it follows that $n - t > n/2 + t$. Hence, every honest party resolves ε (the root node) with $\mathcal{RT}^r(\varepsilon) = v$ by applying the IT resolve rule. $\square$

```

revised Debatepk(v) run by party p:
1 : // k denotes the current round when Debate is invoked in the Cloture Votes
2 : // v denotes the initial value
3 : Initialization : // create root nodes ε for both  $\mathcal{IT}_p^k$  and  $\mathcal{RT}_p^k$ 
4 :  $\mathcal{IT}_p^k(\varepsilon) \leftarrow v$ ;  $\mathcal{RT}_p^k(\varepsilon) \leftarrow \perp$ ;  $M_p^k \leftarrow U_p^k \leftarrow \{\varepsilon\}$ 
5 : // M: set of nodes whose values need to be received
6 : // U: set of nodes whose values are undecided and child nodes are required
7 :  $\mathcal{OUT}_p^k \leftarrow \{(\varepsilon, p, v)\}$ 
8 : for r = k to t + 1 : // the protocol runs t - k + 2 rounds
9 : // send-round r
10 : Send  $\mathcal{OUT}_p^r$  to q ∈ P
11 :  $M_p^k \leftarrow \bigcup_{\sigma \in U_p^k, q \in \mathcal{P} \setminus [\sigma]} \{\sigma q\}$ 
12 : // receive-round r
13 : Receive  $\mathcal{IN}_{q \rightarrow p}^r$  from each party q ∈ P
14 : for every σq ∈ Mpk
15 :   if ∃(σ, q, v) ∈  $\mathcal{IN}_{q \rightarrow p}^r$  :
16 :      $\mathcal{IT}_p^k(\sigma q) \leftarrow v$ 
17 :   elseif r = k ∧  $\nexists(\varepsilon, q, \cdot) \in \mathcal{IN}_{q \rightarrow p}^r$  :
18 :      $\mathcal{IT}_p^k(q) \leftarrow 0$ 
19 :   elseif  $\nexists(\sigma, q, \cdot) \in \mathcal{IN}_{q \rightarrow p}^r$  :
20 :      $\mathcal{IT}_p^k(\sigma q) \leftarrow \mathcal{IT}_p^k(\sigma)$  // nodes without messages follow their parents' values
21 : // fault masking
22 : for σq ∈ Mpk such that q ∈  $\mathcal{F}_p$  :  $\mathcal{IT}_p^k(\sigma q) \leftarrow 0$ 
23 : apply Fault Detection
24 : while  $\mathcal{F}_p \rightarrow \mathcal{F}_p \cup \{q\}$  // repeat fault masking for new Byzantine parties
25 :   for σq ∈ Mpk :  $\mathcal{IT}_p^k(\sigma q) \leftarrow 0$ 
26 : apply Resolve Rules
27 : // early stopping
28 : if  $\mathcal{RT}_p^k(\varepsilon) \neq \perp$  :
29 :   return  $\mathcal{RT}_p^k(\varepsilon)$ 
30 : for σ such that  $\mathcal{RT}_p^k(\sigma) \neq \perp$  :
31 :   remove all descendants of σ from Upk
32 :  $U_p^k \leftarrow \bigcup_{\sigma \in U_p^k, q \in \mathcal{P} \setminus [\sigma]} \{\sigma q\}$ 
33 :  $\mathcal{OUT}_p^{r+1} \leftarrow \bigcup_{\sigma \in U_p^k} \{(\sigma, p, \mathcal{IT}_p^k(\sigma))\}$  // send messages for unresolved nodes only

```

Fig. 7. The *revised* Debate protocol. The different parts with the original Debate protocol [BG93] are highlighted in blue. This is an early-stopping EIG protocol with exponential message complexity that guarantees $\min\{f + 2(r), t + 1(r)\}$ termination, provided $t < n/4$.

```

revised ClotureVotes( $v_p$ ) run by party  $p$  with initial value  $v_p$ :
1 : Initialization :
2 :  $S \leftarrow 2n^2$  // the threshold of the message size
3 :  $v \leftarrow v_p$  // the initial value of  $\text{Debate}_p^1$ 
4 : for  $r = 1$  to  $t + 1$  :
5 :   // send-round
6 :   start  $\text{Debate}_p^r(v)$ 
7 :   Process all ongoing Debate by the end of send-round
8 :   // receive-round
9 :   Process all ongoing Debate by the end of receive-round
10 :  if  $\exists k \leq r$  such that  $\mathcal{RT}_p^k(\varepsilon) = 1$  in  $\text{Debate}_p^k$  :
11 :    return 1
12 :  elseif  $|\mathcal{F}_p| + 1 < r \vee r = t + 1$  :
13 :    return 0 // now we don't need "I am done" message
14 :  if  $\exists k \leq r$  such that  $|U_p^k| \geq S$  in  $\text{Debate}_p^k$  :
15 :    //  $U_p^k$  denotes the set of nodes whose values are undecided in  $\text{Debate}_p^k$ 
16 :    // now the vote is triggered by  $U$  instead of  $M$ 
17 :     $v \leftarrow 1$  // now 1 means vote for Cloture
18 :  else :  $v \leftarrow 0$ 

```

Fig. 8. The *revised* Cloture Votes protocol. The different parts with the original Cloture Votes protocol [BG93] are highlighted in blue. This is an early-stopping Consensus protocol with polynomial message complexity that guarantees $\min\{f + 2(r), t + 1(r)\}$ termination, provided $t < n/4$.

Lemma 5. *Let p and q be honest parties, and suppose that σ is an unreliable node. For $1 \leq r \leq t$, if p sends OUT_p^r containing (σ, p, v) , then q resolves σp with $\mathcal{RT}_q(\sigma p) = v$ by round $r + 1$.*

Proof. In round r , all honest parties that have not closed σ and have not terminated the entire cloture Vote protocol record $\mathcal{IT}(\sigma p) = v$. In round $r + 1$, it is possible that an honest party x sends OUT_x^{r+1} without $(\sigma p, x, \cdot)$ because x has already resolved an ancestor node of σp or has already terminated the Cloture Vote protocol. Nevertheless, since all honest parties that have not closed σ and have not terminated have $\mathcal{IT}(\sigma p) = v$, they record $\mathcal{IT}(\sigma p x) = \mathcal{IT}(\sigma p) = v$. Consequently, from q 's view, there exist distinct parties $x_1, \dots, x_{n-t-1}$ such that $\mathcal{IT}_q(\sigma p x_i) = v$. This follows either from messages sent by x_i or from the new rule in the revised Debate protocol. Since $n > 4t$, it follows that $n - t - 1 > n/2 + t - |\sigma p|$, and therefore q resolves σp with $\mathcal{RT}_q(\sigma p) = v$ by applying the IT resolve rule. $\square$

Lemma 6. *Let p and q be honest parties. For $1 \leq r \leq t$, suppose that p resolves an unreliable node σ using the IT resolve rule with $\mathcal{RT}_p(\sigma) = v$ in round r . Assume that no honest party resolves any ancestor of σ by round $r - 1$, no honest party has terminated by round $r - 1$, and q has not terminated by round r . Then q resolves σ with $\mathcal{RT}_q(\sigma) = v$ by round $r + 1$.*

Proof. In round r , since no honest party has resolved any ancestor of σ by round $r - 1$ and no honest party has terminated by round $r - 1$, every honest party sends a message containing $(\sigma, \cdot, \cdot)$. Because σ is an unreliable node, at most $t - |\sigma|$ unreliable nodes are included in the set $\{\sigma x | x \in \mathcal{P} \setminus [\sigma]\}$. Since p has more than $n/2 + t - |\sigma|$ parties x_i , such that $\mathcal{IT}_p(\sigma x_i) = v$, it follows that at least $n/2$ honest parties send messages containing $(\sigma, \cdot, v)$. Consequently, fewer than $n/2$ honest parties can send messages containing $(\sigma, \cdot, v')$ for any $v' \neq v$. Therefore, q has at most $n/2 + t - |\sigma|$ parties x_j such that $\mathcal{IT}_q(\sigma x_j) = v'$, and hence q cannot resolve σ with $\mathcal{RT}_q(\sigma) = v'$ via the IT resolve rule in round r .

In round $r + 1$, by Lemma 5, more than $n/2$ reliable nodes are resolved with $\mathcal{RT}(\sigma x_i) = v$. Hence, q also resolves σ with $\mathcal{RT}_q(\sigma) = v$ by applying the RT resolve rule. Suppose, for contradiction, that q resolves σ with $\mathcal{RT}_q(\sigma) = v'$ via the RT resolve rule. Then there is at least one reliable node σx such that $\mathcal{RT}_q(\sigma x) = v'$ even though x sent OUT_x^r containing (σ, x, v) . This contradicts Lemma 5. $\square$

Lemma 7. (*Liveness*) *Let p and q be honest parties. For $1 \leq r \leq t$, suppose that p resolves an unreliable node σ with $\mathcal{RT}_p(\sigma) = v$ in round r . Assume that no honest party resolves any ancestor of σ by round $r - 1$, no honest party has terminated by round $r - 1$, and q has not terminated by round r . Then q resolves σ with $\mathcal{RT}_q(\sigma) = v$ by round $r + 1$.*

Proof. If p resolves σ via the IT resolve rule, then Lemma 6 proves the claim. If q resolves σ via the IT resolve rule with $\mathcal{RT}_q(\sigma) = v' \neq v$, then by Lemma 6, it is impossible for p to resolve σ with $\mathcal{RT}_p(\sigma) = v$.

Hence, assume that p resolves σ via the RT resolve rule while q does not apply the IT resolve rule to σ . This implies that a majority of nodes σx_i for $x_i \in \mathcal{P} \setminus [\sigma]$ satisfies $\mathcal{RT}_p(\sigma x_i) = v$. If p resolves all such σx_i via the IT resolve rule, Lemma 6 ensures q also has $\mathcal{RT}_q(\sigma x_i) = v$ in the next round and therefore resolves σ using the RT resolve rule. Otherwise, some child nodes σx_j are resolved by p via the RT resolve rule. The same argument then applies recursively to their descendant nodes. By induction, all relevant descendant nodes are eventually resolved by p via the IT resolve rule, and Lemma 6 proves the claim. $\square$

Lemma 8. *For $1 \leq r \leq t$ and $k \leq r$, suppose that an honest party p determines the output value v in Debate_p^k in round r . If no honest party has terminated by round $r - 1$, and an honest party q has not terminated by round r , then q also determines the output value v in Debate_q^k by round $r + 1$.*

Proof. If p determines the output value v in Debate_p^k , then $\mathcal{RT}_p^k(\varepsilon) = v$ in Debate_p^k . By Lemma 7, if q has not terminated by round r , q has $\mathcal{RT}_q^k(\varepsilon) = v$ by round $r + 1$ and thus determines the output value v in Debate_q^k . $\square$

Lemma 9. *If $|\mathcal{F}_p| + 1 < r$ for an honest party p at the end of round r , then $\mathcal{RT}_p^1(\varepsilon) \neq \perp$ in Debate_p^1 by round r , unless p has already terminated with decision value 1.*

Proof. Note that Byzantine party cannot be detected in round 1. If $|\mathcal{F}_p| + 1 < r$ in round r , then there must exist a specific round $1 < k \leq r$ in which p does not detect any new Byzantine party. If a node σx is not resolved in $\mathcal{RT}_p^1$ by round $|\sigma x| + 1$, then p put x in $\mathcal{F}_p$ according to the fault detection rule. Therefore, this implies that p resolves all nodes σ with $|\sigma| = k - 1$ in Debate_p^1 by round $k \leq r$. Furthermore, the condition of the RT resolve rule implies that if all child nodes of σ are resolved, then σ is also resolved by the RT resolve rule. By induction, this leads to $\mathcal{RT}_p^1(\varepsilon) \neq \perp$.

We consider the following cases.

- Case 1: p has not terminated by the end of round k .
In this case, by round $k \leq r$, p resolves all nodes σ with $|\sigma| = k - 1$ in Debate_p^1 , which implies that $\mathcal{RT}_p^1(\varepsilon) \neq \perp$. (Note that p may not terminate in round k , since there is no guarantee that $\mathcal{RT}_p^1(\varepsilon) = 1$ or $|\mathcal{F}_p| + 1 < k$ in round k .)
- Case 2: p has terminated before round k because $\mathcal{RT}_p^i(\varepsilon) = 1$ for some Debate^i with $i < k$ (while $\mathcal{RT}_p^1(\varepsilon) = \perp$).
In this case, p terminates with decision value 1, as claimed in the lemma.
- Case 3: p has terminated with decision value 0 before round k , with $\mathcal{RT}_p^1(\varepsilon) = \perp$.
This is impossible. To maintain $\mathcal{RT}_p^1(\varepsilon) = \perp$ in every round before k , at least one Byzantine party must corrupt a node in each round from 1 to $k - 1$. Consequently, $|\mathcal{F}_p|$ must increase by at least one in each round from 2 to k , implying that $|\mathcal{F}_p| + 1 \geq k'$ for every round $k' < k$. $\square$

Lemma 10. *Let $a \in \mathbb{R}^+$. If $|U_p^k|$ for an honest party p in Debate^k reaches an^2 at the end of round r and no honest party has terminated by round $r - 2$, then at least $\lfloor 4a \rfloor$ nodes are fully corrupted in Debate^k in round $r - 2$.*

Proof. Suppose that no node is fully corrupted in round $r - 2$. Then all nodes σ with $|\sigma| = r - k - 1$ in Debate^k are resolved by some honest parties in round $r - 1$. Since no honest party has terminated by round $r - 2$, by Lemma 6, p resolve all these nodes by round r , and thus $|U_p^k| = 0$ at the end of round r .

Now assume that b nodes are fully corrupted in round $r - 2$. For each such node, fewer than $n/4$ child nodes can be corrupted, since $n > 4t$. Consequently, the number of unresolved nodes in round r can reach approximately $bn/4$. At the end of round r , $|U_p^k|$ becomes roughly $bn^2/4$ when accounting for all child nodes.

Therefore, if $|U_p^k|$ in Debate^k reaches an^2 at the end of round r , at least $\lfloor 4a \rfloor$ nodes must have been fully corrupted in Debate^k in round $r - 2$. $\square$

Lemma 11. *If $|U_p^k|$ of an honest party p in Debate^k reaches $2^m n^2$ at the end of round r and no honest party has terminated by round $r - 2$, then at least one Byzantine party must be globally detected in each round from $k + 1$ to $r - 1$; over these rounds, all honest parties detect $r + m - k + 1$ Byzantine parties.*

Proof. By Lemma 10, if $|U_p^k|$ in Debate^k reaches $2^m n^2$ at the end of round r , then at least 2^{m+2} nodes must have been fully corrupted in Debate^k in round $r - 2$. Unless at least one Byzantine party fully corrupts a node in Debate^k in each round from k to $r - 2$, those nodes are resolved by some honest parties, and p also resolves all such nodes by Lemma 7. In that case, $|U_p^k|$ in Debate^k becomes 0 by round r . Therefore, at least one Byzantine party must fully corrupt a node in Debate^k in each round from k to $r - 2$. Consequently, all honest parties detect at least one Byzantine party in each round from $k + 1$ to $r - 1$, yielding at least $r - k - 1$ detected Byzantine parties in total.

Suppose that a_i Byzantine parties fully corrupt the nodes as many as possible in each round $k \leq i \leq r - 2$. This behavior results in $\prod_{i=k}^{r-2} a_i$ fully corrupted nodes in round $r - 2$. As established above, $a_i \geq 1$. By the assumption, $\prod_{i=k}^{r-2} a_i \geq 2^{m+2}$. Note that $a_i \leq 2^{a_i - 1}$ for all positive integer a_i . One also observes that

$$\prod_{i=k}^{r-2} a_i \leq 2^{\sum_{i=k}^{r-2} (a_i - 1)}.$$

Therefore,

$$2^{m+2} \leq \prod_{i=k}^{r-2} a_i \leq 2^{\sum_{i=k}^{r-2} (a_i - 1)},$$

which yields $m + 2 \leq \sum_{i=k}^{r-2} (a_i - 1) = -r + k + 1 + \sum_{i=k}^{r-2} a_i$. Consequently, the number of globally detected Byzantine parties by round $r - 1$ is at least

$$\sum_{i=k}^{r-2} a_i \geq r + m - k + 1.$$

This concludes the proof. $\square$

Lemma 12. *Let r be the first round in which an honest party votes 1 in Debate_p^r . Then, no honest party has terminated by round $r - 2$, and at least r Byzantine parties have been globally detected by round $r - 2$.*

Proof. Since r is the first round in which an honest party votes 1, it follows that in every Debate^k with $1 < k < r$, all honest parties vote 0. (If some honest parties abstain because they have already terminated, their abstention is treated as a vote of 0, so the claim still holds.) Consequently, by Lemma 4, all honest parties output 0 for every such Debate^k , and thus $|U^k| = 0$ in each of them at the end of round $r - 1$.

Therefore, if an honest party p votes 1 in round r , it must be that $|U_p^1|$ in Debate^1 reaches $2n^2$ at the end of round $r - 1$. This requires that at least one Byzantine party fully corrupts a node in every round from 1 to $r - 3$; otherwise, $|U_p^1|$ in Debate^1 would become 0 at the end of round $r - 1$. Hence, no honest party has terminated by round $r - 2$, since in every round $i \leq r - 2$ we must have $|\mathcal{F}_p| + 1 \geq i$, and no Debate outputs 1.

Moreover, if an honest party p votes 1 in round r , it implies that $|U_p^1| \geq 2n^2$ at the end of round $r - 1$. Since no honest party has terminated by round $r - 2$, by Lemma 11, all honest parties have detected r Byzantine parties by round $r - 2$. $\square$

Lemma 13. *If an honest party p votes 1 in Debate_p^r , then no honest party has terminated by round $r - 2$, and at least r Byzantine parties have been globally detected by round $r - 2$.*

Proof. Let q be the first honest party that determines the output value 1 in Debate_q^k with $k < r - 1$ in round $r' < r - 1$. Then by Lemma 8, p also terminates with decision value 1 in Debate_p^k by round $r' + 1$, unless some honest party terminates with decision value 0 by round $r' - 1$. In that case, p would not vote 1 in Debate_p^r , which contradicts the assumption. Therefore, no honest party can terminate with decision value 1 by round $r - 2$ unless some honest party terminates with decision value 0 by round $r - 3$.

If r is the first round in which any honest party votes 1 in some Debate , then the claim follows directly from Lemma 12. Now suppose that a is the first round in which any honest party votes 1, with $a < r$. By Lemma 12, no honest party has terminated by round $a - 2$, and there are a Byzantine parties that have been globally detected by that round. Furthermore, every Debate^k with $1 < k < a$ outputs 0 in its starting round.

Since p votes 1 in round r , $|U_p|$ reaches the threshold $2n^2$ at the end of round $r - 1$. Depending on the Debate_p in which this threshold is reached, we consider three cases.

- Case 1: $|U_p^1|$ in Debate_p^1 reaches $2n^2$ at the end of round $r - 1$.
In this case, at least one Byzantine party must fully corrupt a node in each round from $a - 2$ to $r - 3$. Consequently, $|\mathcal{F}_p| + 1 \geq i$ must hold in each round $a - 1 \leq i \leq r - 2$, preventing any honest party from terminating with decision value 0 by round $r - 2$. Consequently, no honest party can terminate with decision value 1 by round $r - 2$, as established above. Furthermore, this ensures that all honest parties detect at least r Byzantine parties by round $r - 2$.
- Case 2: $|U_p^a|$ in Debate_p^a reaches $2n^2$.
In this case, at least one Byzantine party must fully corrupt a node in each round from a to $r - 3$. Thus, $|\mathcal{F}|$ for all honest parties increases by at least one in each round from $a + 1$ to $r - 2$. Since all honest parties have already detected a Byzantine parties by round $a - 2$, they might terminate with decision

value 0 starting from round $a + 2$. However, this cannot happen, since all honest parties must observe an increase in $|\mathcal{F}|$ in round $a + 1$, and the termination condition no longer holds. Hence, no honest party can terminate with decision value 0 by round $r - 2$. Consequently, no honest party can terminate with decision value 1 by round $r - 2$. Therefore, by Lemma 11, all honest parties detect an additional $r - a + 1$ Byzantine parties from round $a + 1$ to $r - 2$. Consequently, all honest parties detect at least $r + 1$ Byzantine parties by round $r - 2$.

- Case 3: $|U_p^b| \geq 2n^2$ in Debate_p^b for some $a < b < r$.

For this to occur, at least one honest party must vote 1 in Debate^b in round b (Lemma 4). If this is due to the large value of $|U^1|$ in Debate^1 , then by Lemma 11 no honest party can terminate by round $b - 2$, and b Byzantine parties are globally detected, as argued above (this situation is identical to Case 1 with b in place of r). Furthermore, at least one Byzantine party must fully corrupt a node in each round from b to $r - 3$ in Debate^b . Hence, no honest party can terminate by round $r - 2$, and at least $r + 1$ Byzantine parties are globally detected (this situation is analogous to Case 2 with b in place of a).

Suppose that the vote of 1 in round b is instead caused by the large value of $|U^c|$ in some Debate^c with $a \leq c < b < r$. Then, at least one honest party must vote 1 in Debate^c in round c . If this vote is caused by the large value of $|U^1|$ in Debate^1 then by Lemma 11 no honest party can terminate by round $c - 2$, and c Byzantine parties are globally detected (this situation is identical to Case 1 with c in place of r). Furthermore, at least one Byzantine party must fully corrupt a node in each round from c to $b - 3$ in Debate^c . Hence, no honest party can terminate by round $b - 2$, and at least $b + 1$ Byzantine parties are globally detected. Moreover, at least one Byzantine party must fully corrupt a node in each round from b to $r - 3$ in Debate^b . Therefore, no honest party can terminate by round $r - 2$, and at least $r + 2$ Byzantine parties are globally detected.

More generally, the vote of 1 in round c may be caused by a large value of $|U^d|$ for some $d < c$. Applying the same reasoning recursively, we conclude that in all cases no honest party can terminate by round $r - 2$, and at least $r + 1$ Byzantine parties are globally detected by that round.

Thus, in all cases, the claim holds. $\square$

Lemma 14. *Let p be the first honest party that terminates with decision value 1. For $1 \leq r \leq t$, if p terminates with decision value 1 in round r , then another honest party q also terminates with decision value 1 by round $r + 1$.*

Proof. The decision value 1 is triggered only when $\mathcal{RT}_p^k(\varepsilon) = 1$ in Debate_p^k for some $k \leq r$. If no honest party has terminated by round $r - 1$ and q has not terminated in round r , then by Lemma 8, q determines $\mathcal{RT}_q^k(\varepsilon) = 1$ in round $r + 1$ and also terminate with decision value 1. Therefore, it suffices to consider the case in which q terminates by r .

Since p is the first party terminate with decision value 1, no honest party has terminated with decision value 1 by round $r - 1$. Furthermore, if q terminate with decision value 1 in round r , then the claim holds. Hence, we only need to consider the case in which q terminates with decision value 0 by round r .

Suppose p terminates with decision value 1 due to $\mathcal{RT}_p^1(\varepsilon) = 1$ in Debate_p^1 , while q terminates with decision value 0. This means that q terminates in some round r' because $|\mathcal{F}_q| + 1 < r'$. However, by Lemma 9, q must observe the output value 1 of Debate^1 by round r' , and thus cannot terminate with decision value 0.

Now consider the case where Debate^k outputs 1 for $k \neq 1$. This implies that at least one honest party votes 1 in round k . By Lemma 13, no honest party has terminated by round $k - 2$, and all honest parties have detected k Byzantine parties by that round.

- If no Byzantine party fully corrupts a node in Debate^k in round k , then some honest parties resolve all nodes in Debate^k in round $k + 1$, and q also outputs 1 by round $k + 2$. Note that q cannot terminate with decision value 0 earlier than round $k + 2$, since q has already detected k Byzantine parties in round $k - 2$ and $|\mathcal{F}_q| + 1 \geq k + 1$ holds. Therefore, q must observe the output of Debate^k and terminates with decision value 1.

- If Byzantine parties fully corrupts at least one node in Debate^k in round k , then q detects an additional Byzantine party in round $k + 1$. Therefore, termination with decision value 0 must be postponed, and q will eventually observe the output of Debate^k and terminate with decision value 1.

□

Lemma 15. *If an honest party p satisfies $|\mathcal{F}_p| + 1 < r$ in round r , then either every Debate_p^k for $1 \leq k \leq r$ has produced an output by round r , or p has already terminated with decision value 1.*

Proof. Lemma 9 guarantees that p must observe the output of Debate_p^1 unless it has already terminated with decision value 1. Therefore, it suffices to consider Debate^k for $1 < k \leq r$.

Case 1: No honest party votes 1 in any Debate^k for $1 < k \leq r$. In this case, all such Debate^k output 0 in their starting round k , and the claim holds.

Case 2: Some honest parties vote 1 in Debate^a and p terminates with decision value 0. By Lemma 13, no honest party has terminated by round $a - 2$, and by that round, all honest parties have detected a Byzantine parties. Hence, no honest party can terminate with decision value 0 until round $a + 2$, which implies $r \geq a + 2$. Therefore, no honest party votes 1 in either Debate^{r-1} or Debate^r ; otherwise, by Lemma 13, at least $r - 1$ or r Byzantine parties would have been detected, and the condition $|\mathcal{F}_p| + 1 < r$ would not hold in round r . Thus, both debates output 0 in their starting round (Lemma 4).

From round $a + 2$, p may terminate with decision value 0. If no Byzantine party fully corrupts a node in Debate^a , then Debate_p^a outputs by round $a + 2$. If Byzantine parties fully corrupt a node in Debate^a in round a , p detects an additional Byzantine party in round $a + 1$, thereby delaying termination. In this case, p will eventually observe the output of Debate^a before terminating with decision value 0.

Case 3: Some honest parties vote 1 in Debate^a and p terminates with decision value 1, as stated in the Lemma. □

Lemma 16. *Let p be the first honest party that terminates with decision value 0. Then another honest party q will eventually terminate with decision value 0.*

Proof. If p terminates with decision value 0 in round r , then $|\mathcal{F}_p| + 1 < r$. By Lemma 15, p must have observed the outputs of all Debate^k for $1 \leq k \leq r$. Moreover, all such Debate^k output 0, since p terminates with decision value 0.

Since p is the first party that terminates with decision value 0, no honest party terminates with decision value 0 by round $r - 1$. Furthermore, no honest party terminates with decision value 1 by round $r - 1$, since all Debate_p^k output 0 for $1 \leq k \leq r$. Therefore, by Lemma 8, q also outputs 0 in all Debate_q^k for $1 \leq k \leq r$.

If q were to terminate with decision value 1, then some Debate^a with $a \geq r + 1$ must output 1. However, by Lemma 13, if some honest parties vote 1 in Debate^a , then no honest party can terminate until round $a - 2 \geq r - 1$, and all honest parties must detect a Byzantine parties. This implies that no honest party could terminate with decision value 0 before round $a + 2 \geq r + 3$, contradicting the assumption that p terminates with decision value 0 in round r .

Therefore, no honest party votes 1 in any Debate^a with $a \geq r + 1$. Since the number of Byzantine parties is finite, q eventually terminates with decision value 0. □

Lemma 17. *If an honest party p outputs v in Debate^k in round $t + 1$ and no honest party has terminated by round $t - 1$, then any other honest party q also outputs v in Debate^k .*

Proof. If some honest parties output in Debate^k by round t , then by Lemma 8, the output is consistent. Therefore, it suffices to consider the case where no honest party determines the output in Debate^k by round t .

Analysis of Debate¹. If no honest party determines its output by round t , then Byzantine parties must have fully corrupted at least one node σ in Debate^1 in round $t - 1$. If σ contains any honest party, then by Lemma 5, all honest parties resolve the ancestor node of σ by round t . Hence, σ must consist solely of Byzantine parties. Again by Lemma 5, all honest child nodes of σ are resolved in round $t + 1$. Therefore, only the unreliable node σx remains. Since σx contains exactly t Byzantine parties, all of its child nodes are honest. Their values are consistent, so all honest parties resolve σx consistently. Thus, the output of Debate^1 is consistent.

Analysis of Debate^k for $1 < k < t + 1$. If all honest parties vote 0 in round k , then Debate^k outputs immediately in its starting round. Therefore, suppose at least one honest party votes 1 in round k . By Lemma 13, all honest parties must have detected k Byzantine parties by round $k - 2$ and hence by the time Debate^k is invoked. Since a globally detected Byzantine party cannot corrupt further nodes, fully corrupted nodes can only appear up to round $t - 1$ in Debate^k . Thus, every node σ with $|\sigma| = t - k + 1$ in Debate^k is either honest or masked. These nodes are consistently resolved in round $t + 1$, so the output of Debate^k is also consistent.

Analysis of Debate^{t+1}. If some honest parties vote 1 in round $t + 1$, then by Lemma 13, all honest parties must have detected $t + 1$ Byzantine parties, which is impossible. Therefore, all honest parties vote 0 in round $t + 1$, and Debate^{t+1} consistently outputs 0 in round $t + 1$. $\square$

Lemma 18. *If $f < t$, all honest parties terminate by round $f + 2$ with a consistent decision value.*

Proof. If no honest party has terminated with 1 by round $f + 2$, all honest parties satisfy $|F| + 1 < f + 2$, since there are only f Byzantine parties. Therefore, all honest parties terminate with decision value 0 by round $f + 2$.

Assume that some honest parties have already terminated with decision value 1 by round $f + 2$. Then, by Lemma 16, no honest party terminate with decision value 0. If they have terminated by round $f + 1$, then by Lemma 14, all honest parties terminate with decision value 1 by round $f + 2$.

It remains to analyze the case where no honest party has terminated with decision value 1 by round $f + 1$.

Case 1: Debate¹ outputs 1. If no honest party determines its output by round $f + 1$, then Byzantine parties must have fully corrupted at least one node σ in Debate^1 in round f . If σ contains any honest party, then by Lemma 5, all honest parties resolve the ancestor node of σ by round $f + 1$. Hence, σ must consist solely of Byzantine parties. Since σ contains exactly f Byzantine parties, all of its child nodes are the reliable node. By Lemma 5, all these nodes are resolved consistently in round $f + 2$. Thus, all honest parties terminate with decision value 1 in round $f + 2$.

Case 2: Debate^k outputs 1 for $1 < k \leq f$. If all honest parties vote 0 in round k , then Debate^k outputs immediately in its starting round. Therefore, suppose that at least one honest party votes 1 in round k . By Lemma 13, all honest parties must have detected k Byzantine parties by the time Debate^k is invoked, leaving $f - k$ Byzantine parties undetected. Recall that any globally detected Byzantine party cannot corrupt further nodes. Therefore, even if one Byzantine party fully corrupts a node in each round from k to $f - 1$, fully corrupted nodes can appear only up to round $f - 1$ in Debate^k . Thus, every node σ with $|\sigma| = f - k + 1$ in Debate^k is either honest or masked. These nodes are consistently resolved in round $f + 1$, so all honest parties terminate with decision value 1 in round $f + 1$.

Case 3: Debate^k outputs 1 for $k \in \{f + 1, f + 2\}$. By Lemma 13, all honest parties must have detected at least $f + 1$ Byzantine parties, which conflicts the assumption that there are only f Byzantine parties. Therefore, all honest parties must vote 0 in round $f + 1$ and $f + 2$, and both Debate^{f+1} and Debate^{f+2} consistently output 0 in their starting rounds (Lemma 4). $\square$

Lemma 19. *All honest parties terminate by round $t + 1$, with a consistent decision value across all honest parties.*

Proof. If some honest parties terminate by round t , then Lemmas 14 and 16 prove the claim.

Suppose instead that no honest party terminates by round t . Then all Debate^k for $1 \leq k \leq t+1$ produce outputs by round $t+1$. By Lemmas 8 and 17, these outputs are consistent. Hence, the decision value is consistent in round $t+1$. $\square$

Lemma 20. *The message complexity is $O(n^4)$.*

Proof. Note that there can be at most $t+1 = O(n)$ calls for Debate for each party in a single protocol execution.

First, assume that $|U_p^k| < n^3$ for all Debate^k for p . Then the maximum message size is trivially bounded by $O(n^4)$.

Now suppose $|U_p^k| \geq n^3$ for some Debate^k at the end of round r . This implies that no honest party has terminated by round $r-2$. Let q be the first party that terminates. If q terminates with decision value 1 by round $r-2$, then by Lemma 14, p terminates by round $r-1$, and thus $|U_p^k| \geq n^3$ cannot hold. If q terminates with decision value 0 in some round $r' \leq r-2$, then no honest party can vote 1 from round $r'+1$; otherwise, all honest parties must have detected at least $r'+1$ Byzantine parties by round $r'-1$, which contradicts the assumption that q terminate with decision value 0 in round r' . Furthermore, by Lemma 15, q already resolves every $\mathcal{RT}_q^{k'} = 0$ for $1 \leq k' \leq r'$ by round r' . Consequently, by Lemma 8, p also resolves every $\mathcal{RT}_p^{k'} = 0$ by round $r'+1$, resulting in $|U_p^{k'}| = 0$, which contradicts the assumption that $|U_p^k| \geq n^3$ for Debate^k in round r .

Since no honest party has terminated by round $r-2$, by Lemma 10, Byzantine parties must fully corrupt at least $4n$ nodes in round $r-2$. Consequently, every honest party has at least $4n$ unresolved nodes in round $r-1$. At the end of round $r-1$, $|U^k|$ equals the number of child nodes of these unresolved nodes, which is approximately $4n^2$. Hence, all honest parties vote 1, and they terminate with decision value 1 in round r . Since the protocol has not terminated by round $r-1$, $|U_p^k| < n^3$ at the end of round $r-1$. Therefore, the worst-case message size of p for Debate^k is $O(n^3)$ in round r . Furthermore, at most $O(n)$ such Debate can exist in round r . Hence, the maximum message size of p is upper bounded by $O(n^4)$. $\square$

Now we are ready to prove the security of the revised Cloture Votes protocol (Figs. 7 and 8).

Proof (of Theorem 6). The protocol's validity directly follows from Lemma 4, while consistency by $\min\{f+2(r), t+1(r)\}$ is ensured by Lemmas 18 and 19. The message complexity of $O(n^4)$ is proved by Lemma 20. $\square$

8 $f+2(r)$ Early-Stopping Broadcast Protocol for $t < n/2$

In this section, we propose a Broadcast protocol that terminates in $\min\{f+2(r), t+1(r)\}$ in the honest majority setting.

Throughout this section, we assume $n > 2t$. Let s be the sender, and $v_s \in \mathcal{D}$ be the input value of the sender. Note that we consider multi-value Broadcast protocols, and $\mathcal{D}$ is not necessarily to be $\{0,1\}$. Furthermore, the output value satisfies $v \in D \cup \{\text{Null}\}$, where Null explicitly indicates that all honest parties agree that the sender is Byzantine. Consequently, Null may be replaced by any other predetermined value, such as 0, as in other BA protocols.

The detailed protocol is illustrated in Fig. 10 and Fig. 11 with the output decision algorithm shown in Fig. 12. Theorem 7 states that the protocol achieves early-stopping Broadcast.

Theorem 7. *The protocol in Fig. 10 and Fig. 11 is a Broadcast protocol that guarantees termination by $\min\{f+2(r), t+1(r)\}$, provided that $n > 2t$.*

Remark 4. Any signature-based BA scheme is vulnerable in principle to attacks that attempt to forge honest parties' signatures. Furthermore, as in [ELP25], our protocol requires each party to sign every projection. Since a single message may contain exponentially many projections, one may wonder whether the aggregate forgery probability could become non-negligible. This issue is avoided by requiring honest parties to accept a

message only if all projection-signature pairs contained in the message are valid. As a result, the adversary's success probability can be bounded by the probability of a single successful forgery in each round, which is negligible. Accordingly, in the proof of the theorem in Section 8.4, we disregard forgery attempts and restrict attention to executions in which no successful forgery occurs.

8.1 Additional Preliminaries

8.1.1 Signature Chains. Digital signatures [GMR88] are massively used to provide authenticity and integrity of messages. Let (sk, vk) be a pair of signing key and verification key of a signature scheme Σ . The signing algorithm that inputs sk and message m and returns a signature sig , which, can be publicly verified by the verification algorithm with vk . We require Σ to have correctness, unforgeability, and consistency.

Notably, in this work we consider information-theoretic signatures [HSZI00, SHZI02], where the above properties hold even against computationally unbounded adversaries. This is motivated by the fact that our Broadcast protocol requires exponentially large messages and therefore incurs exponential running time. We refer to Appendix A for formal definitions of information-theoretic signatures and the corresponding complexity analysis in our protocol.

For any $p \in \mathcal{P}$, let $\text{sig}_p(m)$ be a signature of a message¹⁵ m signed by p . For any $v_p \in \mathcal{D}$, we denote

$$\langle \varepsilon, p, (v_p, 1) \rangle := (\langle \varepsilon, p, (v_p, 1) \rangle, \text{sig}_p(\langle \varepsilon, p, (v_p, 1) \rangle)).$$

Similarly, for any $r > 1$ and $\sigma \in \bigcup_{i=1}^{r-1} [P]^i$, we denote

$$\langle \sigma, p, (\text{Null}, r) \rangle := (\langle \sigma, p, (\text{Null}, r) \rangle, \text{sig}_p(\langle \sigma, p, (\text{Null}, r) \rangle)).$$

Consequently, for any $r > k \geq 1$, $\sigma = p_{i_1} \dots p_{i_{r-1}} \in \bigcup_{i=1}^{r-1} [P]^i$, $p \notin [\sigma]$ and $v \in \mathcal{D} \cup \{\text{Null}\}$ such that $k = 1 \Rightarrow v \in \mathcal{D}$, we recursively define

$$\begin{aligned} \langle p_{i_1} \dots p_{i_k}, p_{i_{k+1}}, (v, k) \rangle &:= \left(\langle p_{i_1} \dots p_{i_{k-1}}, p_{i_k}, (v, k) \rangle, \text{sig}_{p_{i_{k+1}}}(\langle p_{i_1} \dots p_{i_{k-1}}, p_{i_k}, (v, k) \rangle) \right), \\ &\dots \\ \langle p_{i_1} \dots p_{i_{r-2}}, p_{i_{r-1}}, (v, k) \rangle &:= \left(\langle p_{i_1} \dots p_{i_{r-3}}, p_{i_{r-2}}, (v, k) \rangle, \text{sig}_{p_{i_{r-1}}}(\langle p_{i_1} \dots p_{i_{r-3}}, p_{i_{r-2}}, (v, k) \rangle) \right), \\ \langle p_{i_1} \dots p_{i_{r-1}}, p, (v, k) \rangle &:= (\langle p_{i_1} \dots p_{i_{r-2}}, p_{i_{r-1}}, (v, k) \rangle, \text{sig}_p(\langle p_{i_1} \dots p_{i_{r-2}}, p_{i_{r-1}}, (v, k) \rangle)). \end{aligned}$$

8.1.2 Message Format. We consider an EIG-like protocol where parties can add signatures to their messages. Similarly to the basic EIG protocol, for each round r , each party $p \in \mathcal{P}$ is supposed to send to every party an ordered set

$$\widehat{\text{OUT}}_p^r := \left\{ \langle (\mathcal{P} \setminus \{p\})^{r-1}[1], p, v_{p,1}^r \rangle, \dots, \langle (\mathcal{P} \setminus \{p\})^{r-1}[(|\mathcal{P} \setminus \{p\}|)_{r-1}], p, v_{p, (|\mathcal{P} \setminus \{p\}|)_{r-1}}^r \rangle \right\}.$$

In the same way, we can define OUT_p^r and projections accordingly, as well as $\mathcal{IN}_{p \rightarrow q}^r$.

We say $\langle p_{i_1} \dots p_{i_{r-1}}, p_{i_r}, (v, k) \rangle \in \mathcal{IN}_{p_{i_r} \rightarrow p}^r$ is valid if $\langle p_{i_1} \dots p_{i_{r-1}}, p_{i_r}, (v, k) \rangle$ can be decomposed recursively as described above into $r - k + 1$ corresponding signatures from p_{i_k} to p_{i_r} , and all such signatures are valid with respect to the verification keys of $p_{i_k}, \dots, p_{i_r}$.

In our protocol described below, we also require that a valid projection $\langle p_{i_1} \dots p_{i_{r-2}}, p_{i_{r-1}}, (v, k) \rangle$ should have $p_{i_1} = s$, and

1. $(v, k) \in \mathcal{D} \times \{1\}$, or

¹⁵ In all our other descriptions of protocols, a party sends a single message to all parties in each round. However, for defining signature chains, a message denotes the input to a signature scheme. Note that a message of a signature scheme is actually an element, i.e., a projection, of a message of a party in the protocols.

2. $(v, k) \in \{\text{Null}\} \times \mathbb{Z}^+ \setminus \{1\}$.

Note that s can put their valid signature only in round 1, while other parties can put their valid signature only in round $r > 1$. We also require that a party accept a message only if all projection-signature pairs in the message are valid, to avoid non-negligible forgery attacks.

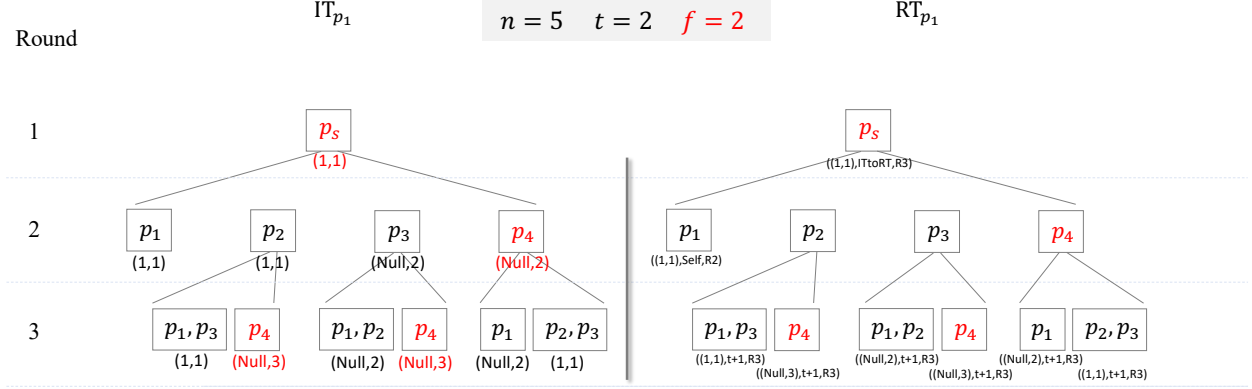


Fig. 9. A toy example-execution: views of $\mathcal{IT}_{p_1}$ and $\mathcal{RT}_{p_1}$ of honest party p_1 after an execution where two parties p_s and p_4 are Byzantine.

8.1.3 Tree Format. Instead of using the same notation of $\mathcal{IT}$ and $\mathcal{RT}$, we extend them to contain information about the round when the signature chain is initiated, i.e., the round when the first signature in the chain is signed. We show a toy example-execution in Fig. 9.

8.2 Description of Rules

The protocol described in Fig. 11 employs the following rules: the Resolve Rules, the Fault Detection Rule, and the Output Decision Rule.

8.2.1 IT Resolve Rules

A party p applies the following $\mathcal{IT}$ resolve rules in round r :

1. **Self-Resolve rule**
For $r \geq 1$, $k \geq 1$, $\sigma \in [\mathcal{P}]^{r-1}$, and $v \in \mathcal{D} \cup \{\text{Null}\}$, if $\mathcal{IT}_p(\sigma p) = (v, k)$, then $\mathcal{RT}_p(\sigma p) := (v, k)$.
2. **Early IT to RT rule**
 - (a) For $r \geq 2$, $k \geq 1$, $\sigma \in [\mathcal{P}]^{r-1}$, and $v \in \mathcal{D} \cup \{\text{Null}\}$, if
 - i. $\mathcal{IT}_p(\sigma) = (v, k)$,
 - ii. $\mathcal{IT}_p(\sigma x) = (v, k)$, or $\mathcal{IT}_p(\sigma x) = \perp$ for all $x \in \mathcal{P} \setminus [\sigma]$,
then $\mathcal{RT}_p(\sigma) := (v, k)$.
 - (b) For $r \geq 2$, $k \geq 1$, $\sigma \in [\mathcal{P}]^{r-1}$, and $v \in \mathcal{D} \cup \{\text{Null}\}$, if
 - i. $\mathcal{IT}_p(\sigma) = \perp$,
 - ii. $\mathcal{IT}_p(\sigma x) = (\text{Null}, |\sigma| + 1)$, or $\mathcal{IT}_p(\sigma x) = \perp$ for all $x \in \mathcal{P} \setminus [\sigma]$,
then $\mathcal{RT}_p(\sigma) := (\text{Null}, |\sigma| + 1)$.
3. **IT to RT rule**
 - (a) For $r \geq 3$, $k \geq 1$, $\sigma \in [\mathcal{P}]^{r-3}$, $v \in \mathcal{D} \cup \{\text{Null}\}$, and $x \in \mathcal{P}$, if

The protocol run by party p in round 1:

```

1 : Initialization :
2 :  $M \leftarrow U \leftarrow \{s\}$  //  $s$  is the designated sender
3 : for  $r = 1$  to  $t + 1$  :
4 :    $OUT_p^r \leftarrow \emptyset$ 
5 :   if  $p = s$  :  $OUT_p^1 \leftarrow \{\langle \varepsilon, p, (v_s, 1) \rangle\}$ 
6 :   // send-round 1
7 :   Send  $OUT_p^1$  to  $q \in \mathcal{P}$ 
8 :   if  $p = s$  : return  $v_s$ 
9 :   // receive-round 1
10 :  Receive  $IN_{s \rightarrow p}^1$  from  $s$ 
11 :  if  $\exists \langle \varepsilon, s, (v, 1) \rangle \in IN_{s \rightarrow p}^1$  :
12 :     $IT_p(s) \leftarrow (v, 1)$ ;  $OUT_p^2 \leftarrow \{\langle s, p, (v, 1) \rangle\}$ 
13 :  else :
14 :     $IT_p(s) \leftarrow \perp$ ;  $OUT_p^2 \leftarrow \{\langle s, p, (\text{Null}, 2) \rangle\}$ 

```

Fig. 10. $\min\{f + 2(r), t + 1(r)\}$ early-stopping Byzantine Broadcast protocol tolerating $t < n/2$ Byzantine parties: round 1. It is implicitly assumed that every received messages passes the signature and structure verification before the update of IT_p based on it, as otherwise the corresponding value in IT_p will be set as $\perp$.

- i. $x \in \mathcal{H}_p$,
- ii. $IT_p(\sigma x) = (v, k) \wedge k \leq |\sigma x|$,
- iii. there exist distinct $y_1, \dots, y_{n-t-1} \in \mathcal{H}_p \setminus [\sigma x]$ such that $IT_p(\sigma x y_i) = (v, k)$ or $IT_p(\sigma x y_i) = \perp$ for all $i \in [1..n-t-1]$,
then σx is called a (v, k) -candidate.
- (b) For $r \geq 3$, $k \geq 1$, $\sigma \in [\mathcal{P}]^{r-3}$, $v \in \mathcal{D} \cup \{\text{Null}\}$, and $x, y \in \mathcal{P}$, if
 - i. σx is a (v, k) -candidate,
 - ii. $y \in \mathcal{H}_p \setminus [\sigma x]$,
 - iii. $IT_p(\sigma x y) = (v, k)$,
 - iv. there exist distinct $z_1, \dots, z_{n-t-2} \in \mathcal{H}_p \setminus [\sigma x y]$ such that $IT_p(\sigma x y z_i) = (v, k)$ or $IT_p(\sigma x y z_i) = \perp$ for all $i \in [1..n-t-2]$,
then $\sigma x y$ is called an *active σx -supporter*.
- (c) For $r \geq 3$, $k \geq 1$, $\sigma \in [\mathcal{P}]^{r-3}$, $v \in \mathcal{D} \cup \{\text{Null}\}$, and $x \in \mathcal{P}$, if
 - i. σx is a (v, k) -candidate,
 - ii. $IT_p(\sigma x p) = (v, k)$,
then $\sigma x p$ is called an *active σx -supporter*.
- (d) For $r \geq 3$, $k \geq 1$, $\sigma \in [\mathcal{P}]^{r-3}$, $v \in \mathcal{D} \cup \{\text{Null}\}$, and $x, y \in \mathcal{P}$, if
 - i. σx is a (v, k) -candidate,
 - ii. $y \in \mathcal{H}_p \setminus [\sigma x]$,
 - iii. $IT_p(\sigma x y) = \perp$,
 - iv. there exist distinct $z_1, \dots, z_{n-|\sigma|-2} \in \mathcal{P} \setminus [\sigma x y]$ such that $IT_p(\sigma x y z_i) = (\text{Null}, |\sigma x| + 2)$ or $IT_p(\sigma x y z_i) = \perp$ for all $i \in [1..n-|\sigma|-2]$,
then $\sigma x y$ is called a *silent σx -supporter*.
- (e) For $r \geq 3$, $k \geq 1$, $\sigma \in [\mathcal{P}]^{r-3}$, $v \in \mathcal{D} \cup \{\text{Null}\}$, and $x \in \mathcal{P}$, if σx is a (v, k) -candidate and there exist $n - t - 1$ distinct (either active or silent) σx -supporters, then $\mathcal{RT}_p(\sigma x) := (v, k)$.

The protocol run by party p for round $2 \leq r \leq t + 1$:

```

1 : for  $r = 2$  to  $t + 1$ 
2 :   // send-round  $r$ 
3 :   Send  $OUT_p^r$  to  $q \in \mathcal{P}$ 
4 :    $M \leftarrow \bigcup_{\sigma \in U, q \in \mathcal{P} \setminus [\sigma]} \{\sigma q\}$ 
5 :   // receive-round  $r$ 
6 :   Receive  $IN_{q \rightarrow p}^r$  from each party  $q \in \mathcal{P}$ 
7 :   for every  $\sigma q \in M$  :
8 :     if  $\exists (\sigma, q, (v, k)) \in IN_{q \rightarrow p}^r \wedge q \notin \mathcal{F}_p$  :
9 :        $IT_p(\sigma q) \leftarrow (v, k)$ 
10 :    else :  $IT_p(\sigma q) \leftarrow \perp$ 
11 :    apply Fault Detection Rule to update  $\mathcal{F}_p$ 
12 :    apply IT Resolve Rules to update  $\mathcal{RT}_p[r]$ ,  $\mathcal{RT}_p[r - 1]$ , and  $\mathcal{RT}_p[r - 2]$ 
13 :    if  $r = t + 1$  : apply Round  $t + 1$  Rule to update  $\mathcal{RT}_p[t + 1]$ 
14 :    apply RT Resolve Rule to update  $\mathcal{RT}_p[r - 1], \dots, \mathcal{RT}_p[1]$ 
15 :     $U \leftarrow \bigcup_{\sigma \in U, q \in \mathcal{P} \setminus [\sigma]} \{\sigma q\}$ 
16 :    for  $\sigma$  such that  $\mathcal{RT}_p(\sigma) \neq \perp$  :
17 :      remove all descendants of  $\sigma$  from  $U$ 
18 :    if  $U = \emptyset$  :
19 :      apply Output Decision Rule to get the output  $v$ 
20 :      return  $v$ 
21 :    for every  $\sigma \in U$  :
22 :      if  $IT_p(\sigma) = (v, k)$  :  $OUT_p^{r+1} \leftarrow OUT_p^{r+1} \cup \{(\sigma, p, (v, k))\}$ 
23 :      elseif  $IT_p(\sigma) = \perp$  :  $OUT_p^{r+1} \leftarrow OUT_p^{r+1} \cup \{(\sigma, p, (\text{Null}, r + 1))\}$ 

```

Fig. 11. $\min\{f + 2(r), t + 1(r)\}$ early-stopping Byzantine Broadcast protocol tolerating $t < n/2$ Byzantine parties: rounds 2 to $t + 1$. It is implicitly assumed that every received messages passes the signature and structure verification before the update of IT_p based on it, as otherwise the corresponding value in IT_p will be set as $\perp$. Here $\mathcal{RT}_p[r]$ denotes the r -th level of $\mathcal{RT}_p$. If no value is assigned to σ in $\mathcal{RT}_p$, then $\mathcal{RT}_p(\sigma) = \perp$. The output decision algorithm is shown in Fig. 12.

8.2.2 Round $t + 1$ Rule

A party p applies the following *round $t + 1$ rule*:

1. Round $t + 1$ rule
 For $k \geq 1$, $\sigma \in [\mathcal{P}]^{t+1}$, and $v \in \mathcal{D} \cup \{\text{Null}\}$,
 (a) if $\mathcal{IT}_p(\sigma) = (v, k)$, then $\mathcal{RT}_p(\sigma) := (v, k)$.
 (b) if $\mathcal{IT}_p(\sigma) = \perp$, then $\mathcal{RT}_p(\sigma) := (\text{Null}, t + 2)$.

8.2.3 RT Resolve Rule

A party p applies the following *RT resolve rule*, which is executed bottom-up in $\mathcal{RT}_p$:

1. RT Resolve Rule
 For $r \geq 3$, $k \geq 1$, $1 \leq r' \leq r - 1$, $\sigma \in [\mathcal{P}]^{r'}$, and $v \in \mathcal{D} \cup \{\text{Null}\}$, if
 (a) $k \leq |\sigma| + 1$,
 (b) there exist distinct $x_1, \dots, x_{n-t} \in \mathcal{P} \setminus [\sigma]$ such that $\mathcal{RT}_p(\sigma x_i) = (v, k)$ for all $i \in [1..n - t]$,
 then $\mathcal{RT}_p(\sigma) := (v, k)$.

8.2.4 Fault Detection Rule

A party p applies the following fault detection rules to update $\mathcal{F}_p$ in round r .

1. Double-Sign Detection Rule
 For $k, k' \geq 1$, $\sigma, \sigma' \in [\mathcal{P}]^*$, $v, v' \in \mathcal{D} \cup \{\text{Null}\}$, and $x \in \mathcal{P}$, if
 (a) $\mathcal{IT}_p(\sigma x) = (v, k) \neq \perp$,
 (b) there exists a descendant $\sigma x \sigma'$ such that $\mathcal{IT}_p(\sigma x \sigma') = (v', k') \neq \perp$,
 (c) $(v', k') \neq (v, k)$,
 (d) $k' \leq |\sigma x|$,
 then add x to $\mathcal{F}_p$.
 Note: It is sufficient to check nodes with $|\sigma'| \leq 2$ in this protocol. In other words, it is sufficient to check the child and grandchild nodes of σx .

8.2.5 Output Decision Rule

The output decision algorithm is shown in Fig. 12.

8.3 Protocol Structure

We first give an intuitive explanation of the structure of the protocol before the detailed proof:

MAINTAINING LOCAL INFORMATION. Every honest party p maintains two local datasets, M and U , which are initialized to contain only the root node s (s denotes the sender). The elements of these datasets are nodes. U denotes the set of nodes none of whose ancestors have been resolved by p ; therefore, p should send information about which message it received from those nodes. On the other hand, M denotes the set of nodes whose information p should record in $\mathcal{IT}_p$. p needs to record information only about the descendant nodes of unresolved nodes. Therefore, at the beginning of every round, p sets M to the set of child nodes of U (Line 4 in Figure 11). Furthermore, at the end of every round, p sets U to the set of child nodes of the current U and removes all descendants of resolved nodes from U , so as to maintain the invariant that U contains only the nodes none of whose ancestors have been resolved by p (Lines 15–17 in Figure 11).

ISSUE AND RECORD THE MESSAGES. In round 1, the sender s has some input value $v_s \in \mathcal{D}$. s signs the message “ $\varepsilon, s, (v_s, 1)$ ” and sends $\langle \varepsilon, s, (v_s, 1) \rangle := (\varepsilon, s, (v_s, 1), \text{sig}_s(\varepsilon, s, (v_s, 1)))$. If an honest party p receives $\langle \varepsilon, s, (v, 1) \rangle$, then p records $\mathcal{IT}_p(s) := (v, 1)$ and adds p ’s signature to send $\langle s, p, (v, 1) \rangle$ in round 2.

The Output Decision Algorithm run by party p in round r :

```

1 : if  $\mathcal{RT}_p(s) = (v, k)$  :
2 :   return  $v$ 
3 : else :
4 :   for  $i = r - 1, \dots, 0$  and  $|\sigma| = i$  and  $\mathcal{RT}_p(\sigma x) \neq \perp$  for all  $x \in \mathcal{P} \setminus [\sigma]$  :
5 :     if  $\mathcal{RT}_p(\sigma) = \perp$  :
6 :       apply RT Resolve Rule to  $\sigma$ 
7 :       if  $\mathcal{RT}_p(\sigma) = \perp$  and  $\exists x$  such that  $\mathcal{RT}_p(\sigma x) = (v, k)$  and  $k \leq |\sigma|$ 
8 :         and  $\exists y$  such that  $\mathcal{RT}_p(\sigma y) = (v', k') \neq (v, k)$  and  $k' \leq |\sigma|$  :
9 :            $\mathcal{RT}_p(\sigma) \leftarrow (\text{Null}, i + 1)$ 
10 :        elseif  $\mathcal{RT}_p(\sigma) = \perp$  and  $\exists x$  such that  $\mathcal{RT}_p(\sigma x) = (v, k)$  and  $k \leq |\sigma|$  :
11 :           $\mathcal{RT}_p(\sigma) \leftarrow (v, k)$ 
12 :        elseif  $\mathcal{RT}_p(\sigma) = \perp$  :
13 :           $\mathcal{RT}_p(\sigma) \leftarrow (\text{Null}, i + 1)$ 
14 :   return  $v$  such that  $\mathcal{RT}_p(s) = (v, k)$ 

```

Fig. 12. The output decision algorithm.

On the other hand, if p does not receive any valid message from s , then p records $\mathcal{IT}_p(s) := \perp$ and initiates a new signature chain to send $\langle s, p, (\text{Null}, 2) \rangle$ in round 2 (Figure 10).

From round 2 onward, every honest party sends and receives messages. If honest party p receives a valid $\langle \sigma, q, (v, k) \rangle$ where $v \in \mathcal{D} \cup \{\text{Null}\}$ in round r , then p records $\mathcal{IT}_p(\sigma q) := (v, k)$ and adds p 's signature to send $\langle \sigma q, p, (v, k) \rangle$ in the next round (Lines 9 and 22 in Figure 11). On the other hand, for each σq , if p does not receive any valid message of $\langle \sigma q, p, \cdot \rangle$ or q is in p 's Byzantine set $\mathcal{F}_p$, then p records $\mathcal{IT}_p(\sigma q) := \perp$ and initiates a new signature chain to send $\langle \sigma q, p, (\text{Null}, r + 1) \rangle$ in the next round (Lines 10 and 23 in Figure 11).

After recording $\mathcal{IT}$, p detects Byzantine parties by applying the Fault Detection Rule. If there exist two conflicting messages signed by q , then q must be a Byzantine party. This detection must precede the application of the Resolve Rules in order to safely resolve nodes via IT to RT rule.

KEY TECHNIQUE 1: ESTABLISHING NODE VALIDITY. The first aim of the protocol is to establish Node Validity: if σ is an unreliable node and an honest party p sends a message containing $\langle \sigma, p, (v, k) \rangle$ in round r , then every honest party resolves σp with $\mathcal{RT}(\sigma p) = (v, k)$ by round $r + 2$ (Lemma 27). To achieve this property, we design IT to RT rule as follows. Suppose that σ is an unreliable node and p sends a message containing $\langle \sigma, p, (v, k) \rangle$ in round r . In this case, there exist $n - t - 1$ honest parties q other than p . If such a q does not resolve any ancestor of σp , then q sends a message containing $\langle \sigma p, q, (v, k) \rangle$, and honest parties record $\mathcal{IT}(\sigma p q) = (v, k)$. Otherwise, q sends a message without $\langle \sigma p, q, (\cdot, \cdot) \rangle$, and honest parties record $\mathcal{IT}(\sigma p q) = \perp$. Since there exist $n - t - 1$ such nodes, σp becomes a (v, k) -candidate.

Suppose that q does not resolve any ancestor of σp . Then, q also sends a message containing $\langle \sigma p, q, (v, k) \rangle$, implying that $\mathcal{IT}(\sigma p q) = (v, k)$. Furthermore, there exist $n - t - 2$ honest parties q_i other than p and q . If such a q_i does not resolve any ancestor of $\sigma p q$, then q_i sends a message containing $\langle \sigma p q, q_i, (v, k) \rangle$, and honest parties record $\mathcal{IT}(\sigma p q q_i) = (v, k)$. Otherwise, q_i sends a message without $\langle \sigma p q, q_i, (\cdot, \cdot) \rangle$, and honest parties record $\mathcal{IT}(\sigma p q q_i) = \perp$. Since there exist $n - t - 2$ such nodes, $\sigma p q$ becomes an active σp -supporter.

Suppose that q has resolved some ancestor of σp . Then, q sends a message without $\langle \sigma p, q, (\cdot, \cdot) \rangle$, implying that $\mathcal{IT}(\sigma p q) = \perp$. Furthermore, since q does not send any valid σp -projection, no party (including Byzantine parties) can generate a valid signature chain. Therefore, if $q_i \in \mathcal{P} \setminus [\sigma p q]$ sends a message containing a valid $\langle \sigma p q, q_i, (\cdot, \cdot) \rangle$, then it must be $\langle \sigma p q, q_i, (\text{Null}, |\sigma p| + 2) \rangle$, and honest parties record $\mathcal{IT}(\sigma p q q_i) = (\text{Null}, |\sigma p| + 2)$. Otherwise, q_i sends a message without $\langle \sigma p q, q_i, (\cdot, \cdot) \rangle$, and honest parties record $\mathcal{IT}(\sigma p q q_i) = \perp$. Since there exist $n - |\sigma p| - 2$ such nodes, $\sigma p q$ becomes a silent σp -supporter.

Consequently, Node Validity is guaranteed by IT to RT rule.

KEY TECHNIQUE 2: BEYOND LIVENESS: EARLY STOPPING VIA PERSISTENCY. Most existing early-stopping EIG-based BA protocols aim to achieve a property known as *Liveness*. Liveness means that if an honest party p resolves a node σ in round r , then all other honest parties also resolve σ consistently within a constant number of rounds. By Liveness, once p resolves σ , it can immediately close σ and no longer needs to consider it. This key property enables early stopping in EIG protocols: if p resolves the root, it can terminate immediately. Although not always explicitly stated, most early-stopping EIG protocols satisfy Liveness [MW94,BG93,ZIP95,GM98,AD15]. However, when $n \leq 3t$, Liveness becomes extremely challenging, although we believe that both early stopping by $f + 2(r)$ and Liveness can still be achieved simultaneously under $2t < n \leq 3t$. Interestingly, if signatures are available, early stopping can be achieved without Liveness. Instead, we introduce *Persistency*: if an honest party p resolves a node σ in round r , then all other honest parties also resolve σ consistently during their output decision process.

To achieve Persistency, our output decision algorithm becomes more complex than any other existing protocols achieving Liveness. When every node at level r has a resolved ancestor or is itself resolved by p in round r (Line 18 in Figure 11), p applies the output decision algorithm.

If the root node (s) is already resolved, p outputs the value of $\mathcal{RT}_p(s)$ (Line 2 in Figure 12). Otherwise, there exists a node σ that is itself unresolved but all of whose child nodes are resolved by p . For such a node, if p can apply RT Resolve Rule, then p resolves σ via RT Resolve Rule (Line 6 in Figure 12). If p cannot apply RT Resolve Rule, then p proceeds to the next step.

If there exist two child nodes resolved with (v, k) and (v', k') , respectively, where $(v, k) \neq (v', k')$ and $k, k' \leq |\sigma|$, then p resolves σ with $\mathcal{RT}_p(\sigma) = (\text{Null}, |\sigma| + 1)$ (Line 9 in Figure 12). This condition implies that the party q with $\sigma = \sigma'q$ has signed two conflicting messages. Furthermore, it is guaranteed that no honest party resolves σ before the output decision process (see Lemma 32 and Theorem 9 for details). Therefore, p nullifies σ by resolving it with $(\text{Null}, |\sigma| + 1)$. Note that no node σ can be resolved with (v, k) such that $k \geq |\sigma| + 2$. Consequently, if σ is resolved with $(\text{Null}, |\sigma| + 1)$, this node cannot affect the resolved value of its parent node, which corresponds to the neutralization of σ .

Otherwise, there are two cases. If there exists a child node resolved with (v, k) where $k \leq |\sigma|$, then p resolves σ with $\mathcal{RT}_p(\sigma) = (v, k)$ (Line 11 in Figure 12). This condition implies that at least one honest party received a message containing $\langle \sigma', q, (v, k) \rangle$ from q with $\sigma = \sigma'q$. Therefore, p resolves σ with $\mathcal{RT}_p(\sigma) = (v, k)$, since it is possible that some honest party has already resolved σ with (v, k) . If there is no child node resolved with (v, k) where $k \leq |\sigma|$, then p resolves σ with $\mathcal{RT}_p(\sigma) = (\text{Null}, |\sigma| + 1)$ (Line 13 in Figure 12). This condition implies that no honest party received a message containing $\langle \sigma', q, (v, k) \rangle$ from q with $\sigma = \sigma'q$. Therefore, p nullifies σ with $\mathcal{RT}_p(\sigma) = (\text{Null}, |\sigma| + 1)$, since no honest party resolved σ before the output decision process.

With this output decision algorithm, the protocol can achieve Persistency (Theorem 9). Consequently, if some honest party resolves a node σ , then every honest party resolves σ consistently. If no honest party resolves a node σ , then every honest party can resolve σ consistently, since the output decision algorithm resolves σ based on the resolved values of its child nodes, which are themselves resolved consistently. During the output decision process, every honest party resolves the unresolved nodes from the deepest ones up to the root node. Finally, every honest party resolves the root node consistently, and $\mathcal{RT}(s)$ is the output value of the protocol (Line 14 in Figure 12).

8.4 Proof of Theorem 7

We provide the following lemmas before proving Theorem 7. As mentioned earlier, all proofs in this section are “informal”, since we do not consider forgery attempts, which only succeed with negligible probability.

Lemma 21. *Let p be an honest party and $IT_p(\sigma q) = (v, k)$. If $k \geq 2$, then $v = \text{Null}$. If p sends OUT_p^r containing $\langle \sigma, p, (v, k) \rangle$ with $k \geq 2$, then $v = \text{Null}$.*

Proof. If there is $\langle \sigma, q, (v, k) \rangle$ in $\mathcal{IN}_{q \rightarrow p}^r$ with $k \geq 2$ and $v \in \mathcal{D}$, this projection is invalid since there is no valid signature signed by s . Consequently, $IT_p(\sigma q) := \perp$. Therefore, if $IT_p(\sigma q) = (v, k)$ with $k \geq 2$, it must be that $v = \text{Null}$. Moreover, if p sends OUT_p^r containing $\langle \sigma, p, (v, k) \rangle$ with $k \geq 2$, it implies that either $IT_p(\sigma) = (v, k)$ or $IT_p(\sigma) = \perp$; in both cases, it follows that $v = \text{Null}$. $\square$

Lemma 22. *Let p and q be honest parties and $x \in \mathcal{P}$. If p sends OUT_p^r containing $\langle \sigma, p, (v, k) \rangle$, then $\text{IT}_q(\sigma p \sigma' x) \neq (v', k')$ for any $(v', k') \neq (v, k)$ with $k' \leq |\sigma p|$. If p sends OUT_p^r without $\langle \sigma, p, (\cdot, \cdot) \rangle$, then $\text{IT}_q(\sigma p \sigma' x) \neq (v'', k'')$ for any $(v'', k'') \neq (v, k)$ with $|k''| \leq |\sigma p|$.*

Proof. Toward a contradiction, assume that $\text{IT}_q(\sigma p \sigma' x) = (v', k')$ for some (v', k') with $k' \leq |\sigma p|$. Note that q can record a value to $\text{IT}_q(\sigma p \sigma' x)$ only if $\langle \sigma p \sigma', x, (v', k') \rangle$ in $\mathcal{IN}_{x \rightarrow q}^{|\sigma p \sigma' x|}$ can be decomposed recursively into $|\sigma p \sigma' x| - k' + 1$ corresponding signatures on (v', k') . Since $k' \leq |\sigma p|$, one of the required signatures must have been signed by p . However, this leads to a contradiction because p signs on only (v, k) leading to $\langle \sigma, p, (v, k) \rangle$.

Now, consider the case where p sends OUT_p^r without $\langle \sigma, p, (\cdot, \cdot) \rangle$. Toward a contradiction, assume that $\text{IT}_q(\sigma p \sigma' x) = (v'', k'')$ for some (v'', k'') with $k'' \leq |\sigma p|$. Again, for q to record a value to $\text{IT}_q(\sigma p \sigma' x)$, $\langle \sigma p \sigma', x, (v'', k'') \rangle$ in $\mathcal{IN}_{x \rightarrow q}^{|\sigma p \sigma' x|}$ must be decomposed recursively into $|\sigma p \sigma' x| - k'' + 1$ corresponding signatures on (v'', k'') . Since $k'' \leq |\sigma p|$, one of the required signatures must have been signed by p . However, this leads to a contradiction because p did not send $\langle \sigma, p, (\cdot, \cdot) \rangle$. $\square$

Lemma 23. *Let p and q be honest parties, and suppose $p \notin [\sigma]$. If p sends $\text{OUT}_p^{|\sigma|+1}$ without $\langle \sigma, p, (\cdot, \cdot) \rangle$, and q does not resolve any ancestor of σp by round $|\sigma| + 2$, then $\mathcal{RT}_q(\sigma p) = (\text{Null}, |\sigma p| + 1)$ at the end of round $|\sigma| + 2$.*

Proof. Since $\langle \sigma, p, (\cdot, \cdot) \rangle$ is not included in $\text{OUT}_p^{|\sigma|+1}$, no party can produce a valid $\langle \sigma p, \cdot, (v, k) \rangle$ with $k \leq |\sigma p|$. Therefore, for any $x \in \mathcal{P} \setminus [\sigma p]$, it must hold that $\text{IT}_q(\sigma p x) = \perp$ or $\text{IT}_q(\sigma p x) = (\text{Null}, |\sigma p| + 1)$. Consequently, in round $|\sigma| + 2$, q resolves σp with $\mathcal{RT}_q(\sigma p) = (\text{Null}, |\sigma p| + 1)$ via the Early IT to RT rule. $\square$

Lemma 24. *Let p and q be honest parties, and suppose $p \notin [\sigma]$. If p sends $\text{OUT}_p^{|\sigma|+1}$ without $\langle \sigma, p, (\cdot, \cdot) \rangle$, and q terminates without resolving any ancestor of σp , then $\mathcal{RT}_q(\sigma p) = (\text{Null}, |\sigma p| + 1)$.*

Proof. Let $|\sigma p| \leq t$. If q terminates without resolving any ancestor of σp , it implies that q must have terminated after round $|\sigma p| + 1$ since q cannot resolve σp in round $|\sigma p|$, resulting in $U_q \neq \emptyset$. By Lemma 23, this leads to $\mathcal{RT}_q(\sigma p) = (\text{Null}, |\sigma p| + 1)$.

Now consider the case $|\sigma p| = t + 1$. Then, by Round $t + 1$ rule, $\mathcal{RT}_q(\sigma p) = (\text{Null}, |\sigma p| + 1)$. $\square$

Lemma 25. *Let p and q be honest parties. q will not add p to $\mathcal{F}_q$.*

Proof. Let p sends $\text{OUT}_p^{|\sigma|+1}$ containing $\langle \sigma, p, (v, k) \rangle$. By Lemma 22, there is no descendant node $\sigma p \sigma'$ such that $\text{IT}_q(\sigma p \sigma') = (v', k')$ with $k' \leq |\sigma p|$. Therefore, p will not be added to $\mathcal{F}_q$ via the Double-Sign Detection Rule. $\square$

Lemma 26. *Let p and q be honest parties. If p sends $\text{OUT}_p^{|\sigma|+1}$ containing $\langle \sigma, p, (v, k) \rangle$, then $\mathcal{RT}_q(\sigma p) \neq (v', k')$ for any $(v', k') \neq (v, k)$ until q 's termination.*

Proof. If $|\sigma p| = t + 1$, then $\mathcal{RT}_q(\sigma p) = (v, k)$ by Round $t + 1$ rule. Hence, we consider the case where $|\sigma p| \leq t$.

If $p = q$, then $\mathcal{RT}_q(\sigma p) = (v, k)$ by the Self-resolve rule. Thus, we assume $p \neq q$.

Suppose q resolves σp with $\mathcal{RT}_q(\sigma p) = (v', k')$ using either the Early IT to RT rule or the IT to RT rule. This would require that $\text{IT}_q(\sigma p) = (v', k')$, which is impossible.

Now, suppose q resolves σp with $\mathcal{RT}_q(\sigma p) = (v', k')$ using RT resolve rule. This implies that there exist distinct parties $x_1, \dots, x_{n-t} \in \mathcal{P} \setminus [\sigma p]$ such that $\mathcal{RT}_q(\sigma p x_i) = (v', k')$ with $k' \leq |\sigma p| + 1$ for all $i \in [1..n-t]$. Since $n \geq 2t + 1$, there must be at least one honest party x such that $\mathcal{RT}_q(\sigma p x) = (v', k')$

Consider the case where x sends no σp -projection. In order for q to resolve $\sigma p x$ using IT to RT rule, $\langle \sigma p, x, (\cdot, \cdot) \rangle$ is required, which contradicts the assumption. Therefore, q must have used either the Early IT to RT rule or the RT resolve rule. If q applies Early IT to RT rule, then $\mathcal{RT}_q(\sigma p x) = (\text{Null}, |\sigma p| + 2)$, which contradicts the assumption that $k' \leq |\sigma p| + 1$. Suppose q applies the RT resolve rule. Then, by Lemma 22, the child nodes of $\sigma p x$ can produce only projection of the form $\langle \sigma p \sigma', \cdot, (\text{Null}, k'') \rangle$ with $k'' \geq |\sigma p| + 2$. Hence, resolving $\sigma p x$ this way also yields $\mathcal{RT}_q(\sigma p x) = (\text{Null}, |\sigma p| + 2)$, again contradicting $k' \leq |\sigma p| + 1$.

Now consider the case where x sends $\text{OUT}_x^{|\sigma|+2}$ containing $\langle \sigma p, x, (v, k) \rangle$. If $\mathcal{RT}_q(\sigma p x) = (v', k')$, this becomes an inductive instance of the current claim. Thus, if q resolves $\sigma p x$ as (v', k') , it must do so using

the RT resolve rule. In that case, there exists at least one honest party y such that $\mathcal{RT}_q(\sigma p x y) = (v', k')$. By induction, there exists an honest party p' such that $\mathcal{RT}_q(\sigma p \sigma' p') = (v', k')$ where $|\sigma p \sigma' p'| = t + 1$, since $n \geq 2t + 1$. However, this contradicts the Round $t + 1$ rule. $\square$

Remark 5. Note that the node σp referenced in the above corollaries and lemmas is not necessarily a reliable node. Therefore, even if σ includes some honest parties, the statements in the corollaries and lemmas remain valid. In Lemma 26, we showed that if p sends $\text{OUT}_p^{|\sigma|+1}$ containing $\langle \sigma, p, (v, k) \rangle$, then σp may either be resolved with (v, k) or remain unresolved, but it cannot be resolved with a different value, (v', k') . We now show that if σ contains no honest parties, then σp must be resolved with (v, k) .

Lemma 27. *Let p and q be honest parties and suppose σ is an unreliable node. For $1 \leq r \leq t - 1$, if p sends OUT_p^r containing $\langle \sigma, p, (v, k) \rangle$ in round r , and q does not resolve any ancestor of σp by round $r + 2$, then $\mathcal{RT}_q(\sigma p) = (v, k)$ by round $r + 2$.*

Proof. We prove this lemma by observing the behavior of the honest parties in each round.

- Round $|\sigma p|$: Party p resolves σp with (v, k) using Self-resolve rule. All honest parties that do not resolve any ancestor of σp record $\mathcal{IT}(\sigma p) = (v, k)$.
- Round $|\sigma p| + 1$: Since σ consists solely of Byzantine parties, a total of $n - t - 1$ honest parties (excluding p) either send $\langle \sigma p, \cdot, (v, k) \rangle$ or no σp -projection. Thus, there exist distinct parties $y_1, \dots, y_{n-t-1} \in \mathcal{H} \setminus [\sigma p]$ such that $\mathcal{IT}(\sigma p y_i) = (v, k)$ or $\mathcal{IT}(\sigma p y_i) = \perp$ for all $i \in [1..n - t - 1]$. By Lemma 25, p will not be added to $\mathcal{F}$, and consequently, σp becomes a (v, k) -candidate. If q resolves σp in round $|\sigma p| + 1$, it must do so via the Early IT to RT rule. In that case, since $\mathcal{IT}_q(\sigma p) = (v, k)$, q resolves σp with $\mathcal{RT}_q(\sigma p) = (v, k)$.
- Round $|\sigma p| + 2$: If an honest party y sends $\langle \sigma p, y, (v, k) \rangle$, then $n - t - 2$ honest parties (excluding p and y) either send $\langle \sigma p y, \cdot, (v, k) \rangle$ or no $\sigma p y$ -projection. Therefore, there exist distinct parties $z_1, \dots, z_{n-t-2} \in \mathcal{H} \setminus [\sigma p y]$ such that $\mathcal{IT}(\sigma p y z_i) = (v, k)$ or $\mathcal{IT}(\sigma p y z_i) = \perp$ for all $i \in [1..n - t - 2]$. By Lemma 25, y will not be added to $\mathcal{F}$, and $\sigma p y$ becomes an active σp supporter.

If $y = q$, then q becomes an active σp supporter for q .

If y send a message without $\langle \sigma p, y, (\cdot, \cdot) \rangle$, then all its child nodes either send $\langle \sigma p y, \cdot, (\text{Null}, |\sigma p| + 2) \rangle$ or no $\sigma p y$ -projection. By Lemma 25, y will not be added to $\mathcal{F}$, and $\sigma p y$ becomes a silent σp supporter.

Since σp is a (v, k) -candidate, and there are $n - t - 1$ σp -supporters, q resolves σp using IT to RT rule, resulting in $\mathcal{RT}_q(\sigma p) = (v, k)$. $\square$

Theorem 8. (Validity) *Let p and q be honest parties and suppose σ is an unreliable node. If p sends $\text{OUT}_p^{|\sigma|+1}$ containing $\langle \sigma, p, (v, k) \rangle$, and q terminates without resolving any ancestor of σp , then $\mathcal{RT}_q(\sigma p) = (v, k)$ after the output decision process.*

Proof (of Theorem 8). If q terminates after round $|\sigma p| + 2$, then by Lemma 27, $\mathcal{RT}_q(\sigma p) = (v, k)$, and thus the claim holds. If q has terminates by round $|\sigma p| + 1$ without resolving any ancestor of σp , then either q resolves σp or $|\sigma p| \geq t$.

Suppose $|\sigma p| \leq t - 1$ and q resolves σp in round $|\sigma p| + 1$. In this case, q must apply the Early IT to RT rule to σp , and since $\mathcal{IT}_q(\sigma p) = (v, k)$, it follows that $\mathcal{RT}_q(\sigma p) = (v, k)$.

Now consider the case where $|\sigma p| = t$. In round $t + 1$, if there exists an honest party $x \neq p$ that has not resolved any ancestor of σp , then x sends $\langle \sigma p, x, (v, k) \rangle$ in round $t + 1$. By the Round $t + 1$ rule, this ensures $\mathcal{RT}_q(\sigma p x) = (v, k)$. Furthermore, by Lemma 22, no child node $\sigma p y$ can be resolved with $\mathcal{RT}_q(\sigma p y) = (v', k')$ with $k' \leq |\sigma p|$. Therefore, $\mathcal{RT}_q(\sigma p) = (v, k)$ during the output decision process. If, on the other hand, no honest party remains that has not resolved an ancestor node of σp , then p is the only honest party responsible for resolving σp . In this case, p resolves σp using the Self-resolve rule, with $\mathcal{RT}_p(\sigma p) = (v, k)$.

Finally, if $|\sigma p| = t + 1$, then $\mathcal{RT}_q(\sigma p) = (v, k)$ directly by the Round $t + 1$ rule. $\square$

Remark 6. By Theorem 8, if a reliable node σp sends $\langle \sigma, p, (v, k) \rangle$, and honest party q terminates without resolving any ancestor of σp , then $\mathcal{RT}_q(\sigma p) = (v, k)$. However, if there is an honest party within σ , the claim may not hold. In such a case, let $p' \in [\sigma]$ be an honest party. Then, by Theorem 8, the node $\sigma' p' \sqsubseteq \sigma$ must be resolved to the value (v', k') that it sent. Since the ancestor of σp , $\sigma' p'$ must be resolved consistently in $\mathcal{RT}$,

the value of $\mathcal{RT}(\sigma p)$ becomes irrelevant to the final decision. Therefore, it suffices to consider only reliable nodes.

Moreover, if some honest parties have already resolved an ancestor of a reliable node σp , they will not resolve σp , and thus $\mathcal{RT}(\sigma p)$ may be inconsistent. However, in this case, the ancestor of σp must be an unreliable node. We will show that if an unreliable node $\sigma''x$ is resolved in $\mathcal{RT}$, then $\mathcal{RT}(\sigma''x)$ is consistent across all honest parties. As in the above argument, the claim may not hold if an honest party has already resolved an ancestor of $\sigma''x$. Nevertheless, in that situation, the claim holds for the resolved ancestor node, and the value of $\mathcal{RT}(\sigma''x)$ need not be considered, since its ancestor must be resolved consistently in $\mathcal{RT}$.

Lemma 28. *Let p be an honest party, and let σx be an unreliable node. If, before the output decision process, p has $\mathcal{RT}_p(\sigma x) = (v, k)$ with $k \leq |\sigma x|$, then at least one honest party q must have sent $\langle \sigma x, q, (v, k) \rangle$.*

Proof. Since x is a Byzantine party, p does not apply Self-resolve rule to σx . If p applies either the Early IT to RT rule or the IT to RT rule to σx , then it must have $\langle \sigma, x, (v, k) \rangle$ in $\mathcal{IN}_{x \rightarrow p}^{|\sigma x|}$ with $k \leq |\sigma x|$. Consequently, p sends $\langle \sigma x, p, (v, k) \rangle$, and the claim holds.

Now consider the case where p applies the RT resolve rule to σx . This implies that there exist $n-t$ distinct parties $y_1, \dots, y_{n-t}$ such that $\mathcal{RT}_p(\sigma x y_i) = (v, k)$ for all $i \in [1..n-t]$. Since $n \geq 2t+1$, at least one of the nodes $\sigma x q$ must be a reliable node. By Lemma 27, such an honest party q must have sent $\langle \sigma x, q, (v, k) \rangle$, because if q had not sent a message containing $\langle \sigma x, q, (\cdot, \cdot) \rangle$, then by Lemma 23, $\mathcal{RT}_p(\sigma x q) = (\text{Null}, |\sigma x|+2)$, contradicting the assumption that $k \leq |\sigma x|$. Therefore, the claim holds. $\square$

Lemma 29. *Let p and q be honest parties. For $1 \leq r \leq t-1$, if p resolves an unreliable node σx using the Early IT to RT rule with $\mathcal{RT}_p(\sigma x) = (v, k)$ in round r , and no honest party resolves any ancestor of σx by round $r+2$, then $\mathcal{RT}_q(\sigma x) = (v, k)$ by round $r+2$.*

Proof. We prove this lemma by observing the behavior of the honest parties in each round.

- Round $|\sigma x|+1$: Since no honest party resolves any ancestor of σx , all honest parties will send messages containing $\langle \sigma x, \cdot, (\cdot, \cdot) \rangle$. As p resolves σx with $\mathcal{RT}_p(\sigma x) = (v, k)$ using the Early IT to RT rule, all honest parties must have sent $\langle \sigma x, \cdot, (v, k) \rangle$. Therefore, if q resolves σx via the Early IT to RT rule, $\mathcal{RT}_q(\sigma x) = (v, k)$.
- Round $|\sigma x|+2$: Suppose q resolves σx using the IT to RT rule. If $k \leq |\sigma x|$, then since q sent $\langle \sigma x, q, (v, k) \rangle$ corresponding to $\mathcal{IT}_q(\sigma x) = (v, k)$, it follows that $\mathcal{RT}_q(\sigma x) = (v, k)$. On the other hand, if $k = |\sigma x|+1$, then $\mathcal{IT}_q(\sigma x) = \perp$, and thus q cannot apply the IT to RT rule to σx .
- Round $|\sigma x|+3$: By Lemma 27, $n-t$ reliable nodes $\sigma x y_1, \dots, \sigma x y_{n-t}$ are resolved with $\mathcal{RT}_q(\sigma x y_i) = (v, k)$ for all $i \in [1..n-t]$ by this round. Suppose, for contradiction, that q resolves σx via the RT resolve rule with $\mathcal{RT}_q(\sigma x) = (v', k') \neq (v, k)$. Since $n \geq 2t+1$, there must exist at least one reliable node $\sigma x y$ such that $\mathcal{RT}_q(\sigma x y) = (v', k')$, which contradicts Lemma 27. Therefore, q must resolve σx using the RT resolve rule with $\mathcal{RT}_q(\sigma x) = (v, k)$ by round $|\sigma x|+3$. $\square$

Lemma 30. *Let p and q be honest parties. If p resolves an unreliable node σx using the Early IT to RT rule with $\mathcal{RT}_p(\sigma x) = (v, k)$, and no honest party resolves any ancestor of σx before the output decision process, then $\mathcal{RT}_q(\sigma x) = (v, k)$ after the output decision process.*

Proof. By Lemma 29, if q resolves σx before the output decision process, then $\mathcal{RT}_q(\sigma x) = (v, k)$, and the claim holds. Therefore, suppose that q does not resolve σx before the output decision process.

From Lemma 29, we know that q must resolve σx by round $|\sigma x|+3$, which implies that q must terminate by round $|\sigma x|+2$. Since no honest party resolves any ancestor of σx , if q terminates without resolving σx , then one of the following must hold: (1) q resolves all child nodes of σx using the Early IT to RT rule, or (2) $t-1 \leq |\sigma x| \leq t$. Note that $|\sigma x| \neq t+1$ since there are at most t Byzantine parties.

Suppose that $|\sigma x| \leq t-2$ and that q resolves all child nodes of σx in round $|\sigma x|+2$. Since no ancestor of σx is resolved by any honest party, all honest parties must have sent $\langle \sigma x, \cdot, (\cdot, \cdot) \rangle$. Because p resolves σx with $\mathcal{RT}_p(\sigma x) = (v, k)$ using the Early IT to RT rule, every honest party must have sent $\langle \sigma x, \cdot, (v, k) \rangle$. Therefore, when q resolves the $n-t$ reliable nodes $\sigma x y_1, \dots, \sigma x y_{n-t}$, it follows from Theorem 8 that $\mathcal{RT}_q(\sigma x y_i) = (v, k)$

for all $i \in [1..n - t]$. Consequently, q applies the RT resolve rule to σx , yielding $\mathcal{RT}_q(\sigma x) = (v, k)$, and the claim holds.

Now consider the case where $|\sigma x| = t - 1$. Again, since p resolves σx with $\mathcal{RT}_p(\sigma x) = (v, k)$, all honest parties must have sent $\langle \sigma x, \cdot, (v, k) \rangle$. Since q does not resolve σx and, by assumption, no ancestor of σx is resolved, q must send $\langle \sigma x y_i, q, (v, k) \rangle$ for each reliable nodes $\sigma x y_i \neq \sigma x q$. By the Round $t + 1$ rule, this ensures $\mathcal{RT}_q(\sigma x y_i q) = (v, k)$. Moreover, by Lemma 22, no child node of $\sigma x y_i$ can be resolved with $\mathcal{RT}_q(\sigma x y_i z) = (v', k')$ with $k' \leq |\sigma x y_i|$. Hence, by Line 11 in Fig. 12, q resolves $\sigma x y_i$ with $\mathcal{RT}_q(\sigma x y_i) = (v, k)$ during the output decision process. For the node $\sigma x q$, q applies the Self-resolve rule, resulting in $\mathcal{RT}_q(\sigma x q) = (v, k)$. Therefore, q applies the RT resolve rule to σx , obtaining $\mathcal{RT}_q(\sigma x) = (v, k)$, and the claim holds.

Finally, in the case where $|\sigma x| = t$, the Round $t + 1$ rule directly ensures that $\mathcal{RT}_q(\sigma x y_i) = (v, k)$ for $n - t$ reliable nodes $\sigma x y_i$. Hence, q applies the RT resolve rule to σx by Line 6 in Fig. 12, yielding $\mathcal{RT}_q(\sigma x) = (v, k)$, which completes the proof. $\square$

Lemma 31. *Let p and q be honest parties. If p resolves an unreliable node σx using the IT to RT rule with $\mathcal{RT}_p(\sigma x) = (v, k)$, and q also resolves σx before the output decision process, then $\mathcal{RT}_q(\sigma x) = (v, k)$.*

Proof. Since p resolves σx using the IT to RT rule with $\mathcal{RT}_p(\sigma x) = (v, k)$, it must have sent $\langle \sigma x, p, (v, k) \rangle$. Therefore, if q applies the Early IT to RT rule to σx , it follows that $\mathcal{RT}_q(\sigma x) = (v, k)$.

Now, suppose that q resolves σx using the IT to RT rule with $\mathcal{RT}_q(\sigma x) = (v', k') \neq (v, k)$ and $k' \leq |\sigma x|$. Then, q sends $\langle \sigma x, q, (v', k') \rangle$, which causes p to add x to $\mathcal{F}_p$. As a result, σx cannot be the (v, k) -candidate for p , contradicting the assumption that p resolved σx via the IT to RT rule. Therefore, if q applies the IT to RT rule to σx , then $\mathcal{RT}_q(\sigma x) = (v, k)$.

Next, suppose that q resolves σx using the RT resolve rule with $\mathcal{RT}_q(\sigma x) = (v'', k'') \neq (v, k)$ and $k'' \leq |\sigma x| + 1$. This implies that there exist distinct nodes $\sigma x y_1, \dots, \sigma x y_{n-t}$ such that $\mathcal{RT}_q(\sigma x y_i) = (v'', k'')$ for all $i \in [1..n-t]$. Since $n \geq 2t + 1$, at least one node $\sigma x y$ must be a σx -supporter for p , while $\mathcal{RT}_q(\sigma x y) = (v'', k'')$.

Assume $\sigma x y$ is an active σx -supporter. If y is honest, then it is impossible for $\sigma x y$ to support the (v, k) -candidate σx for p , while $\mathcal{RT}_q(\sigma x y) = (v'', k'')$. Hence, y must be Byzantine. Since $\sigma x y$ is an active σx -supporter, p must have received $\langle \sigma x, y, (v, k) \rangle$ from y , and thus sent $\langle \sigma x y, p, (v, k) \rangle$. Consequently, q cannot resolve $\sigma x y$ via the IT resolve rules with $\mathcal{RT}_q(\sigma x y) = (v'', k'')$. Thus, suppose instead that q resolves $\sigma x y$ via the RT resolve rule. Then there exist at least one reliable node $\sigma x y z$ such that $\mathcal{RT}_q(\sigma x y z) = (v'', k'')$. By Lemma 27, $\sigma x y z$ must have sent $\langle \sigma x y, z, (v'', k'') \rangle$, allowing p to observe two conflicting σy -projections—one with (v, k) and the other with (v'', k'') —both signed by y . Hence, p would add y to $\mathcal{F}_p$, implying that $\sigma x y$ cannot be a σx -supporter for p , contradicting the assumption.

Now assume that $\sigma x y$ is a silent σx -supporter, meaning that no honest party received $\langle \sigma x, y, (\cdot, \cdot) \rangle$. If y is honest, then by Lemma 23, $\mathcal{RT}_q(\sigma x y) = (\text{Null}, |\sigma x| + 2)$, contradicting the assumption that $k'' \leq |\sigma x| + 1$. Thus, y must be Byzantine. Since q does not receive $\langle \sigma x, y, (\cdot, \cdot) \rangle$, q cannot resolve $\sigma x y$ via the IT to RT rules. If q resolves $\sigma x y$ using the Early IT to RT rule, then $\mathcal{RT}_q(\sigma x y) = (\text{Null}, |\sigma x| + 2)$, again contradicting $k'' \leq |\sigma x| + 1$. Therefore, suppose that q resolves $\sigma x y$ via the RT resolve rule. Then at least one reliable node $\sigma x y z$ must be resolved with $\mathcal{RT}_q(\sigma x y z) = (v'', k'')$. By Lemma 27, $\sigma x y z$ must have sent $\langle \sigma x y, z, (v'', k'') \rangle$. However, since no honest party received $\langle \sigma x, y, (\cdot, \cdot) \rangle$, z could only have sent $\langle \sigma x y, z, (\text{Null}, |\sigma x| + 2) \rangle$, contradicting $k'' \leq |\sigma x| + 1$. If z does not send $\langle \sigma x y, z, (\cdot, \cdot) \rangle$, then by Lemma 23, $\mathcal{RT}_q(\sigma x y z) = (\text{Null}, |\sigma x| + 3)$, again contradicting $k'' \leq |\sigma x| + 1$.

Therefore, in all cases, if q resolves σx before the output decision process, then $\mathcal{RT}_q(\sigma x) = (v, k)$. $\square$

Lemma 32. *Let p and q be honest parties. If p resolves an unreliable node σx using the IT to RT rule with $\mathcal{RT}_p(\sigma x) = (v, k)$, and no honest party resolves any ancestor of σx before the output decision process, then $\mathcal{RT}_q(\sigma x) = (v, k)$ after the output decision process.*

Proof. By Lemma 31, if q resolves σx before the output decision process, then $\mathcal{RT}_q(\sigma x) = (v, k)$, and the claim holds. Thus, suppose that q does not resolve σx before the output decision process.

Since p resolves σx using the IT to RT rule with $\mathcal{RT}_p(\sigma x) = (v, k)$, it must have sent $\langle \sigma x, p, (v, k) \rangle$. Then, by Theorem 8, every honest party that does not resolve σx must have $\mathcal{RT}(\sigma x p) = (v, k)$.

Now assume that q resolves a child node σxy with $\mathcal{RT}_q(\sigma xy) = (v', k') \neq (v, k)$ and $k' \leq |\sigma x|$. By Lemma 28, at least one honest party p' must have sent $\langle \sigma xy, p', (v', k') \rangle$. In this case, p observes two conflicting σx -projections signed by x . Therefore, p would add x to $\mathcal{F}_p$, contradicting the assumption that σx is a (v, k) -candidate for p . Hence, before the output decision process, q cannot resolve any child node σxy with $\mathcal{RT}_q(\sigma xy) = (v', k') \neq (v, k)$ where $k' \leq |\sigma x|$.

Now suppose that q resolves σx during the output decision process with $\mathcal{RT}_q(\sigma x) = (v'', k'') \neq (v, k)$. This requires that q must resolve at least one child node σxy with $\mathcal{RT}_q(\sigma xy) = (v', k') \neq (v, k)$ and $k' \leq |\sigma x|$; otherwise, by Line 11 in Fig. 12, q would resolve σx with $\mathcal{RT}_q(\sigma x) = (v, k)$, since $\mathcal{RT}(\sigma xp) = (v, k)$ is guaranteed by Theorem 8.

Consider the case where σxy is a reliable node. If σxy does not send $\langle \sigma x, y, (\cdot, \cdot) \rangle$, then by Lemma 24, $\mathcal{RT}_q(\sigma xy) = (\text{Null}, |\sigma x| + 2)$, contradicting $k' \leq |\sigma x|$. If σxy sends $\langle \sigma x, y, (\text{Null}, |\sigma x| + 1) \rangle$, then Theorem 8 implies $\mathcal{RT}_q(\sigma xy) = (\text{Null}, |\sigma x| + 1)$, again contradicting $k' \leq |\sigma x|$. If σxy sends $\langle \sigma x, y, (v, k) \rangle$, then $\mathcal{RT}_q(\sigma xy) = (v, k)$, contradicting $(v', k') \neq (v, k)$. If σxy sends $\langle \sigma x, y, (v', k') \rangle$, then p would add x to $\mathcal{F}_p$, again contradicting the assumption. Hence, σxy must be an unreliable node.

Now suppose that no honest party resolves σx using the Early IT to RT rule. (Otherwise, Lemma 30 implies the claim.) In this case, all honest parties send σx -projections up to round $|\sigma x| + 2$. If the unreliable node σxy sends no σx -projection to any honest party, then all honest parties send $\langle \sigma xy, \cdot, (\text{Null}, |\sigma x| + 2) \rangle$, and by Theorem 8, $\mathcal{RT}_q(\sigma xyz_i) = (\text{Null}, |\sigma x| + 2)$ for all $i \in [1..n - t]$. Consequently, q resolves σxy using the RT resolve rule with $\mathcal{RT}_q(\sigma xy) = (\text{Null}, |\sigma x| + 2)$, contradicting $k' \leq |\sigma x|$.

Otherwise, at least one honest party q' must have received $\langle \sigma x, y, (v'', k'') \rangle$ with $k'' \leq |\sigma x| + 1$, and sent $\langle \sigma xy, q', (v'', k'') \rangle$. If $k'' \leq |\sigma x|$, then p would add x to $\mathcal{F}_p$, contradicting the assumption. Thus, suppose $k'' = |\sigma x| + 1$. Then, by Theorem 8, $\mathcal{RT}_q(\sigma xyq') = (v'', k'')$. It is possible that another node σxyz is resolved with $\mathcal{RT}_q(\sigma xyz) = (v', k')$ during the output decision process. However, since $\mathcal{RT}_q(\sigma xyq') = (v'', k'')$, σxy must be resolved either as $(\text{Null}, |\sigma x| + 2)$ by Line 9 in Fig. 12 or as (v'', k'') by Line 11 in Fig. 12, both of which contradict $k' \leq |\sigma x|$.

Therefore, in all cases, $\mathcal{RT}_q(\sigma x) = (v, k)$. □

Lemma 33. *Let p and q be honest parties. If p resolves an unreliable node σx with $\mathcal{RT}_p(\sigma x) = (v, k)$ where $k \leq |\sigma x| + 1$, no honest party resolves any ancestor of σx , and q also resolves σx before the output decision process, then $\mathcal{RT}_q(\sigma x) = (v, k)$.*

Proof. If either p or q applies the IT resolve rules to σx , then the claim follows directly from Lemmas 29 and 31.

Note that there is no unreliable node σx such that $|\sigma x| = t + 1$. Consequently, neither p nor q applies the Round $t + 1$ rule to σx . Suppose that both p and q apply the RT resolve rule to σx . Since $n \geq 2t + 1$, there exists at least one child node σxy that is used by both parties in resolving σx . If either p or q applies the IT resolve rules to σxy , then by Lemma 29 and 31, we have $\mathcal{RT}_p(\sigma xy) = \mathcal{RT}_q(\sigma xy)$, which implies $\mathcal{RT}_p(\sigma x) = \mathcal{RT}_q(\sigma x)$. Otherwise, both p and q apply the RT resolve rule to σxy .

If σxy is a reliable node, then by Lemma 23 and Lemma 27, it follows that $\mathcal{RT}_p(\sigma xy) = \mathcal{RT}_q(\sigma xy)$, again implying $\mathcal{RT}_p(\sigma x) = \mathcal{RT}_q(\sigma x)$. Thus, assume that σxy is an unreliable node. In this case, both p and q must use a common child node σxyz when resolving σxy . By induction, there exists a common descendant node $\sigma x\sigma'p'$ with $|\sigma x\sigma'p'| = t + 1$ that is used by both parties in the resolution process.

Note that we assume every ancestor of $\sigma x\sigma'p'$ is an unreliable node at each level. Therefore, $\sigma x\sigma'$ must be an unreliable node, which implies that $\sigma x\sigma'p'$ is a reliable node. Consequently, we have $\mathcal{RT}_p(\sigma x\sigma'p') = \mathcal{RT}_q(\sigma x\sigma'p')$, which implies $\mathcal{RT}_p(\sigma x) = \mathcal{RT}_q(\sigma x)$. □

Theorem 9. (Persistencecy) *Let p and q be honest parties. If p resolves an unreliable node σx with $\mathcal{RT}_p(\sigma x) = (v, k)$, and no honest party resolves any ancestor of σx before the output decision process, then $\mathcal{RT}_q(\sigma x) = (v, k)$ after the output decision process.*

Proof. If q resolves σx before the output decision process, then by Lemma 33, $\mathcal{RT}_q(\sigma x) = (v, k)$. Thus, assume q does not resolve σx before the output decision process.

If p applies the IT resolve rules to σx , then the claim follows directly from Lemmas 30 and 32.

Now suppose that p applies the RT resolve rule to σx with $\mathcal{RT}_p(\sigma x) = (v, k)$ where $k \leq |\sigma x| + 1$. If all child nodes σxy_i used in p 's resolution are already resolved by q , then by Lemma 33, we have $\mathcal{RT}_p(\sigma xy_i) = \mathcal{RT}_q(\sigma xy_i)$ for all $i \in [1..n - t]$. Hence, q also applies the RT resolve rule to σx , yielding $\mathcal{RT}_q(\sigma x) = (v, k)$.

Otherwise, there exists a child node σxy used in p 's resolution that is not resolved by q before the output decision process. This means that q terminates without resolving σxy or any of its ancestor nodes. If σxy is a reliable node, then by Theorem 8, we have $\mathcal{RT}_p(\sigma xy) = \mathcal{RT}_q(\sigma xy)$, which implies $\mathcal{RT}_p(\sigma x) = \mathcal{RT}_q(\sigma x)$.

Now consider the case where σxy is an unreliable node. If p applies the IT resolve rules to σxy , then by Lemmas 30 and 32, we again have $\mathcal{RT}_p(\sigma xy) = \mathcal{RT}_q(\sigma xy)$, and hence $\mathcal{RT}_p(\sigma x) = \mathcal{RT}_q(\sigma x)$. Otherwise, p applies the RT resolve rules to σxy . In that case, there exist $n - t$ distinct grandchild nodes σxyz_i used in p 's resolution. If all these nodes are resolved by q before output decision process, then by Lemma 33, q also applies the RT resolve rule to σxy , yielding $\mathcal{RT}_q(\sigma x) = (v, k)$. Otherwise, there exists a grandchild node σxyz used in p 's resolution that is not resolved by q before the output decision process. By induction, there exists a descendant node $\sigma x\sigma'p'$ with $|\sigma x\sigma'p'| = t + 1$ that is used in p 's resolution. Since $\sigma x\sigma'p'$ must be a reliable node, it follows that $\mathcal{RT}_p(\sigma x\sigma'p') = \mathcal{RT}_q(\sigma x\sigma'p')$, which implies $\mathcal{RT}_p(\sigma x) = \mathcal{RT}_q(\sigma x)$. $\square$

Lemma 34. *All honest parties terminate by round $t + 1$, with a consistent output value across all of them.*

Proof. In round $t + 1$, all honest parties that have not yet terminated resolve all $t + 1$ -level nodes using the Round $t + 1$ rules. Consequently, all honest parties terminate by round $t + 1$.

If s is honest, then by Theorem 8, the output value is consistent. Otherwise, if some honest parties resolve s , Theorem 9 ensures that the output value is consistent.

Suppose that no honest party resolves s . By Theorem 8, the $\mathcal{RT}$ values of all reliable nodes sx_i are consistent. If all child nodes of s are resolved by some honest parties, then Theorem 9 implies that the $\mathcal{RT}$ values of all child nodes are consistent across honest parties. Hence, the output value is consistent.

Now consider an unreliable node sx that is not resolved by any honest party. By Theorem 8, the $\mathcal{RT}$ values of all reliable nodes sxy_i are consistent. If all child nodes of sx are resolved by some honest parties, Theorem 9 ensures that the $\mathcal{RT}$ values of all these child nodes are consistent, which implies that $\mathcal{RT}(sx)$, and consequently $\mathcal{RT}(s)$, are consistent.

By induction, there exists an unreliable node σ where $|\sigma| = t$, such that no honest party resolve σ or any of its ancestor nodes. Since all its child nodes are reliable nodes, their $\mathcal{RT}$ values are consistent. Thus, the output value is consistent. $\square$

Lemma 35. *If $f < t$, all honest parties terminate by round $f + 2$, with a consistent output value across all of them.*

Proof. Assume that an honest party q does not terminate at the end of round $f + 2$. This implies that there exist a node σ of depth $f + 1$ such that q does not resolve σ or any of its ancestor nodes. Since $|\sigma| = f + 1$, σ must include at least one honest party p . Let $\sigma'p \sqsubseteq \sigma$ be the shortest prefix such that σ' is an unreliable node.

Suppose that $|\sigma'p| \leq f$. By Lemma 23 and Lemma 27, all honest parties that have not resolved any ancestor of $\sigma'p$ must resolve $\sigma'p$ by round $|\sigma'p| + 2$, contradicting the assumption.

Now consider the case where $|\sigma'p| = f + 1$. In this case, σ' contains all Byzantine parties, so all child nodes of $\sigma'p$ must be honest. If $\sigma'p$ sends $\langle \sigma', p, (v, k) \rangle$, then q observes that all child nodes of $\sigma'p$ either send $\langle \sigma'p, \cdot, (v, k) \rangle$ or no $\sigma'p$ -projection. Thus, q applies the Early IT to RT rule to resolves $\sigma'p$, contradicting the assumption. On the other hand, if $\sigma'p$ sends no σ' -projection, then all child nodes of $\sigma'p$ either send $\langle \sigma'p, \cdot, (\text{Null}, f + 2) \rangle$ or no $\sigma'p$ -projection. Again, q resolves $\sigma'p$ using the Early IT to RT rule, leading to a contradiction. Therefore, all honest parties must terminate by round $f + 2$.

If s is honest, then by Theorem 8, the output value is consistent. Otherwise, if some honest parties resolve s , Theorem 9 ensures that the output value is consistent.

Suppose that no honest party resolves s . By Theorem 8, the $\mathcal{RT}$ values of all reliable nodes sx_i are consistent. If all child nodes of s are resolved by some honest parties, then Theorem 9 implies that the $\mathcal{RT}$ values of all child nodes are consistent across honest parties. Hence, the output value is consistent.

Now consider an unreliable node sx that is not resolved by any honest party. By Theorem 8, the $\mathcal{RT}$ values of all reliable nodes sxy_i are consistent. If all child nodes of sx are resolved by some honest parties, Theorem 9 ensures that the $\mathcal{RT}$ values of all these child nodes are consistent, which implies that $\mathcal{RT}(sx)$, and consequently $\mathcal{RT}(s)$, are consistent.

By induction, there exists an unreliable node σ where $|\sigma| = f$, such that no honest party resolve σ or any of its ancestor nodes. Since all its child nodes are reliable nodes, their $\mathcal{RT}$ values are consistent. Thus, the output value is consistent. $\square$

Now we are ready to prove the security of our Broadcast protocol in Figs. 11 and 12.

Proof (of Theorem 7). The protocol's validity directly follows from Theorem 8, and consistency by $\min\{f + 2(r), t + 1(r)\}$ is ensured by Lemmas 34 and 35. $\square$

9 Conclusion

We introduce a new fine-grained notion of round, which consists of a send-(mini)-round and a receive-(mini)-round, for consistent counting of round complexity (Section 4). Based on this fine-grained round, we find that the known lower bound for the round complexity of BA against crash faults is in fact $f + 2(s)$ while all existing BAs against Byzantine faults are required to have at least $f + 2(r)$ rounds. Thus, we newly prove that the lower bound for the round complexity of BA against Byzantine faults is $f + 2(r)$ (Section 5).

This leaves an open question about the tightness of BA against crash and omission faults. Is $f + 2(r)$ necessary, or is $f + 2(s)$ sufficient? To answer this question, we provide a new Broadcast protocol for any $t < n$, achieving $\min\{f + 2(s), t + 1(r)\}$ round complexity against omission faults (Section 6). Thus, we show that $f + 2(s)$ is the tight round complexity against crash or omission faults.

Furthermore, we prove that if $t > n/2$ and f is sufficiently small, then BA requires at least $f + 3(s)$ (Section 5). This result implies that $f + 3(s)$ may be the tight round complexity for BA in the dishonest-majority setting, although we fail to prove it for large f . This again leaves an open question about the tightness of BA. Is $f + 2(r)$ the tight round complexity for $t < n/2$? A BA protocol achieving $f + 2(r)$ round complexity for $t < n/3$ is already known [BGP92]. However, for $t \geq n/3$, such a protocol is unknown. We provide a new elegant protocol achieving $f + 2(r)$ round complexity for $t < n/2$ (Section 8). Thus, we prove that $f + 2(r)$ is the tight round complexity for $t < n/2$, while $f + 3(s)$ may be the tight round complexity for $t > n/2$.

Some protocols [BG93, MW94] claimed to achieve optimal round complexity ($f + 2$), but their actual round complexity is $f + 3(s)$. This difference is nontrivial. We demonstrate a revision of [BG93] that fixes its round complexity from $f + 3(s)$ to $f + 2(r)$ (Section 7). This revision involves several nontrivial changes to the protocol.

Open questions:

1. What is the lower bound of round complexity for $t = n/2$?
2. Is $f + 3(s)$ the tight round complexity for $t > n/2$?

References

- ACD⁺19. 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.
- AD15. 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.
- ADD⁺19. Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. Synchronous byzantine agreement with expected $O(1)$ rounds, expected $O(n^2)$ communication, and optimal resilience. In Ian Goldberg and Tyler Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 320–334. Springer, Cham, February 2019.

- ADH08. Ittai Abraham, Danny Dolev, and Joseph Y. Halpern. An almost-surely terminating polynomial protocol for asynchronous byzantine agreement with optimal resilience. In Rida A. Bazzi and Boaz Patt-Shamir, editors, *27th ACM PODC*, pages 405–414. ACM, August 2008.
- AS24. Ittai Abraham and Gilad Stern. Early stopping, same but different: Two rounds are needed even in failure-free executions. <https://decentralizedthoughts.github.io/2024-01-28-early-stopping-lower-bounds/>, 2024. Blog post, “Decentralized Thoughts”, Jan. 28, 2024.
- Ben83. Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In Robert L. Probert, Nancy A. Lynch, and Nicola Santoro, editors, *2nd ACM PODC*, pages 27–30. ACM, August 1983.
- BG93. Piotr Berman and Juan A. Garay. Cloture votes: $n/4$ -resilient distributed consensus in $t+1$ rounds. *Math. Syst. Theory*, 26(1):3–19, 1993.
- BGP92. Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Optimal early stopping in distributed consensus (extended abstract). In Adrian Segall and Shmuel Zaks, editors, *Distributed Algorithms, 6th International Workshop, WDAG '92, Haifa, Israel, November 2-4, 1992, Proceedings*, volume 647 of *Lecture Notes in Computer Science*, pages 221–237. Springer, 1992.
- BHMN21. Iddo Bentov, Pavel Hubáček, Tal Moran, and Asaf Nadler. Tortoise and hares consensus: The meshcash framework for incentive-compatible, scalable cryptocurrencies. In Shlomi Dolev, Oded Margalit, Benny Pinkas, and Alexander A. Schwarzmann, editors, *Cyber Security Cryptography and Machine Learning - 5th International Symposium, CSCML 2021, Be'er Sheva, Israel, July 8-9, 2021, Proceedings*, Lecture Notes in Computer Science, pages 114–127. Springer, 2021.
- BNDDS92. Amotz Bar-Noy, Danny Dolev, Cynthia Dwork, and H. Raymond Strong. Shifting gears: changing algorithms on the fly to expedite byzantine agreement. *Inf. Comput.*, 97:205–233, 1992.
- CBS04. Bernadette Charron-Bost and André Schiper. Uniform consensus is harder than consensus. *Journal of Algorithms*, 51(1):15–37, 2004.
- CDL⁺25. Ran Cohen, Jack Doerner, Eysa Lee, Anna Lysyanskaya, and Lawrence Roy. An unstoppable ideal functionality for signatures and a modular analysis of the dolev-strong broadcast. LNCS, pages 675–708. Springer, Cham, November 2025.
- CFG⁺23. Ran Cohen, Pouyan Forghani, Juan A. Garay, Rutvik Patel, and Vassilis Zikas. Concurrent asynchronous byzantine agreement in expected-constant rounds, revisited. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part IV*, volume 14372 of *LNCS*, pages 422–451. Springer, Cham, November / December 2023.
- CLOZ25. Michele Ciampi, Yun Lu, Rafail Ostrovsky, and Vassilis Zikas. Two-tier black-box blockchains and application to instant layer-1 payments. In Zeta Avarikioti and Nicolas Christin, editors, *7th Conference on Advances in Financial Technologies, AFT 2025, Pittsburgh, PA, USA, October 8-10, 2025*, LIPIcs, pages 19:1–19:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.
- CPS20. 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, Cham, May 2020.
- CR93. Ran Canetti and Tal Rabin. Fast asynchronous byzantine agreement with optimal resilience. In *25th ACM STOC*, pages 42–51. ACM Press, May 1993.
- DKLZ25. Giovanni Deligios, Ivana Klasovita, and Chen-Da Liu-Zhang. Asymptotically optimal early termination for dishonest majority broadcast. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part V*, volume 15605 of *LNCS*, pages 116–144. Springer, Cham, May 2025.
- DM86. Cynthia Dwork and Yoram Moses. Knowledge and common knowledge in a byzantine environment I: crash failures. In Joseph Y. Halpern, editor, *Proceedings of the 1st Conference on Theoretical Aspects of Reasoning about Knowledge, Monterey, CA, USA, March 1986*, pages 149–169. Morgan Kaufmann, 1986.
- DRS90. Danny Dolev, Rüdiger Reischuk, and H. Raymond Strong. Early stopping in byzantine agreement. *J. ACM*, 37(4):720–741, 1990.
- DS83. Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM J. COMPUT.*, 12(4):656–666, 1983.
- ELP24. Fatima Elsheimy, Julian Loss, and Charalampos Papamanthou. Early stopping byzantine agreement in $(1 + \epsilon) \cdot f$ rounds. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part VI*, volume 15489 of *LNCS*, pages 398–424. Springer, Singapore, December 2024.
- ELP25. Fatima Elsheimy, Julian Loss, and Charalampos Papamanthou. Towards optimal early stopping agreement protocols. Cryptology ePrint Archive, Report 2025/246, 2025.

- ELT22. Karim Eldefrawy, Julian Loss, and Ben Terner. How byzantine is a send corruption? In Giuseppe Ateniese and Daniele Venturi, editors, *ACNS 2022*, volume 13269 of *LNCS*, pages 684–704. Springer, Cham, June 2022.
- FL82. Michael J Fischer and Nancy A Lynch. A lower bound for the time to assure interactive consistency. *Information processing letters*, 14(4):183–186, 1982.
- FLP85. Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.
- FLZL21. Matthias Fitzi, Chen-Da Liu-Zhang, and Julian Loss. A new way to achieve round-efficient byzantine agreement. In Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen, editors, *40th ACM PODC*, pages 355–362. ACM, July 2021.
- FM88. Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *20th ACM STOC*, pages 148–161. ACM Press, May 1988.
- FN09. Matthias Fitzi and Jesper Buus Nielsen. On the number of synchronous rounds sufficient for authenticated byzantine agreement. In Idit Keidar, editor, *Distributed Computing, 23rd International Symposium, DISC 2009, Elche, Spain, September 23-25, 2009. Proceedings*, Lecture Notes in Computer Science, pages 449–463. Springer, 2009.
- GKKO07. 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.
- GKTZ12. Adam Groce, Jonathan Katz, Aishwarya Thiruvengadam, and Vassilis Zikas. Byzantine agreement with a rational adversary. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *ICALP 2012, Part II*, volume 7392 of *LNCS*, pages 561–572. Springer, Berlin, Heidelberg, July 2012.
- GM98. Juan A. Garay and Yoram Moses. Fully polynomial byzantine agreement for $n > 3t$ processors in $t + 1$ rounds. *SIAM J. COMPUT.*, 27(1):247–290, 1998.
- GMR88. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.
- GP90. Oded Goldreich and Erez Petrank. The best of both worlds: Guaranteeing termination in fast randomized byzantine agreement protocols. *Inf. Process. Lett.*, 36(1):45–49, 1990.
- HSZI00. Goichiro Hanaoka, Junji Shikata, Yuliang Zheng, and Hideki Imai. Unconditionally secure digital signature schemes admitting transferability. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 130–142. Springer, Berlin, Heidelberg, December 2000.
- IKK⁺11. Yuval Ishai, Jonathan Katz, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On achieving the “best of both worlds” in secure multiparty computation. *SIAM J. Comput.*, 40(1):122–141, 2011.
- IOZ14. Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 369–386. Springer, Berlin, Heidelberg, August 2014.
- KK06. 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, Berlin, Heidelberg, August 2006.
- KW20. Pankaj Khanchandani and Roger Wattenhofer. Brief announcement: Byzantine agreement with unknown participants and failures. In Yuval Emek and Christian Cachin, editors, *39th ACM PODC*, pages 178–180. ACM, August 2020.
- LLR02. Yehuda Lindell, Anna Lysyanskaya, and Tal Rabin. On the composition of authenticated byzantine agreement. In *34th ACM STOC*, pages 514–523. ACM Press, May 2002.
- LN24. Julian Loss and Jesper Buus Nielsen. Early stopping for any number of corruptions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part III*, volume 14653 of *LNCS*, pages 457–488. Springer, Cham, May 2024.
- LS23. Julian Loss and Gilad Stern. Zombies and ghosts: Optimal byzantine agreement in the presence of omission faults. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part IV*, volume 14372 of *LNCS*, pages 395–421. Springer, Cham, November / December 2023.
- LSP82. Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.
- MW94. Yoram Moses and Orli Waarts. Coordinated traversal: $(t + 1)$ -round byzantine agreement in polynomial time. *J. Algorithms*, 17:110–156, 1994.
- PCR14. Arpita Patra, Ashish Choudhury, and C. Pandu Rangan. Asynchronous byzantine agreement with optimal resilience. *Distributed Comput.*, 27(2):111–146, 2014.

- PR04. Philippe Raipin Parvédy and Michel Raynal. Optimal early stopping uniform consensus in synchronous systems with process omission failures. In *SPAA 2004*, pages 302–310. ACM, 2004.
- PS18. Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, Lecture Notes in Computer Science, pages 3–33. Springer, 2018.
- PSL80. Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.
- PT86. Kenneth J. Perry and Sam Toueg. Distributed agreement in the presence of processor and communication faults. *IEEE Trans. Software Eng.*, 12(3):477–482, 1986.
- Rab83. Michael O. Rabin. Randomized byzantine generals. In *24th FOCS*, pages 403–409. IEEE Computer Society Press, November 1983.
- SHZI02. Junji Shikata, Goichiro Hanaoka, Yuliang Zheng, and Hideki Imai. Security notions for unconditionally secure signature schemes. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 434–449. Springer, Berlin, Heidelberg, April / May 2002.
- SS11. Colleen Swanson and Douglas R. Stinson. Unconditionally secure signature schemes revisited. In Serge Fehr, editor, *ICITS 11*, volume 6673 of *LNCS*, pages 100–116. Springer, Berlin, Heidelberg, May 2011.
- WXSD20. 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, Cham, November 2020.
- Zam96. Arkady Zamsky. A randomized byzantine agreement protocol with constant expected time and guaranteed termination in optimal (deterministic) time. In James E. Burns and Yoram Moses, editors, *15th ACM PODC*, pages 201–208. ACM, August 1996.
- ZHM09. Vassilis Zikas, Sarah Hauser, and Ueli M. Maurer. Realistic failures in secure multi-party computation. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 274–293. Springer, Berlin, Heidelberg, March 2009.
- ZIP95. Arkady Zamsky, Amos Israeli, and Shlomit S. Pinter. Optimal time byzantine agreement for $t < n/8$ with linear-messages. *Distrib. Comput.*, 9:95–108, 1995.

A Information-Theoretic Signatures

We recall the definition of information-theoretic signatures from [IOZ14], which refines the original definitions in [HSZI00,SHZI02] in a more intuitive way. Let $\mathcal{P}$ denote the set of parties with $|\mathcal{P}| = n$. Let $\ell_M, \ell_S, \ell_V \in \mathbb{N}^+$ be positive integers specifying the length of messages, the upper bound of signing, and the upper bound of verification, respectively.

Definition 10 ($\mathcal{P}$ -Verifiable Signatures). A $\mathcal{P}$ -verifiable signature scheme Σ consists of a triple of randomized algorithms $\Sigma = (\text{Gen}, \text{Sign}, \text{Ver})$.

- $\text{Gen}(1^\lambda, n, \ell_M, \ell_S, \ell_V)$. The key generation algorithm takes parameters $\lambda, n, \ell_M, \ell_S, \ell_V$ as input, and outputs a signing key sk and a vector of verification keys $(vk_1, \dots, vk_n)$.
- $\text{Sign}(sk, m)$. The signing algorithm takes sk and a message $m \in \{0, 1\}^{\ell_M}$ as input, and outputs a signature sig .
- $\text{Ver}(vk_i, m, \text{sig})$. The verification algorithm takes a verification key vk_i , a message m and a signature sig as input, and outputs a bit b indicating the validity of sig w.r.t. vk_i and m .

We require correctness, unforgeability, and consistency of Σ .

Definition 11 (Security of Signatures). Let $\Sigma = (\text{Gen}, \text{Sign}, \text{Ver})$ be a $\mathcal{P}$ -verifiable signature scheme. We say Σ is information-theoretically (ℓ_M, ℓ_S, ℓ_V) -secure if it satisfies the following properties.

- **Correctness (or Completeness).** An honestly generated signature will verify by an (honest) receiver. Formally, for all $i \in [n]$ and all $m \in \{0, 1\}^{\ell_M}$, it holds that

$$\Pr[(sk, vk_1, \dots, vk_n) \leftarrow \text{Gen} : \text{Ver}(vk_i, m, \text{Sign}(sk, m)) = 1] = 1.$$

- **Unforgeability.** An (computationally unbounded) adversary $\mathcal{A}$ cannot forge a signature of a fresh message. Formally, for all adversary $\mathcal{A}$ that queries $\mathcal{O}_{\text{Sign}}$ and $\mathcal{O}_{\text{Ver}}$ at most ℓ_S and ℓ_V times, respectively, any $\mathcal{C} \subsetneq [n]$, it holds that

$$\Pr \left[\begin{array}{l} (sk, vk_1, \dots, vk_n) \leftarrow \text{Gen}(1^\lambda, n, \ell_M, \ell_S, \ell_V) \\ (i^*, m^*, \text{sig}^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Sign}}, \mathcal{O}_{\text{Ver}}}(\{vk_i\}_{i \in \mathcal{C}}) \end{array} : \begin{array}{l} i^* \notin \mathcal{C} \wedge m^* \notin Q_{\text{Sign}} \\ \text{Ver}(vk_{i^*}, m^*, \text{sig}^*) = 1 \end{array} \right] \leq \text{negl}(\lambda).$$

- **Consistency.** An (computationally unbounded) adversary $\mathcal{A}$ cannot create a signature that verifies differently by two parties. Formally, for all adversary $\mathcal{A}$ that queries $\mathcal{O}_{\text{Sign}}$ and $\mathcal{O}_{\text{Ver}}$ at most ℓ_S and ℓ_V times, respectively, any $\mathcal{C} \subsetneq [n]$, it holds that

$$\Pr \left[\begin{array}{l} (sk, vk_1, \dots, vk_n) \leftarrow \text{Gen}(1^\lambda, n, \ell_M, \ell_S, \ell_V) \\ (m^*, \text{sig}^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Sign}}, \mathcal{O}_{\text{Ver}}}(sk, \{vk_i\}_{i \in \mathcal{C}}) \end{array} : \begin{array}{l} \exists i, j \notin \mathcal{C} \text{ such that} \\ \text{Ver}(vk_i, m^*, \text{sig}^*) \neq \text{Ver}(vk_j, m^*, \text{sig}^*) \end{array} \right] \leq \text{negl}(\lambda).$$

- Here $Q_{\text{Sign}}(m)$ is a signing oracle that returns $\text{sig} \leftarrow \text{Sign}(sk, m)$ and adds m into Q_{Sign} , and $\mathcal{O}_{\text{Ver}}(i, m, \text{sig})$ is a verification oracle that returns $b \leftarrow \text{Ver}(vk_i, m, \text{sig})$.

In [SHZI02], Shikata et al. proposed a concrete signature scheme Σ from multi-variate polynomials, where given parameters ℓ_M, ℓ_S, ℓ_V ,

- the signature has a bit-length $O(n\ell_M)$,
- the secret key has a bit-length $O(n\ell_S\ell_M)$,
- the (single) verification key has a bit-length $O(n\ell_M)$,
- the key generation time is $O(n^2\ell_S\ell_M^2)$,
- the signing time $O(n\ell_S\ell_M^2)$,
- the verification time is $O((n + \ell_S)\ell_M^2)$.

Note the signature size exceeds the message size in the scheme of [SHZI02]. However, in our protocol in Section 8, the signature generated in one round is included in the message to be signed in the next round. To accommodate this, we use an independent instantiation of Σ in each round. Concretely, for each round $1 \leq r \leq t + 1$, a party p runs the signing and verification algorithm using keys $(sk_p^r, vk_1^r, \dots, vk_n^r)$, generated by the key generation with parameters $(1^{\lambda_r}, n, \ell_{M,r}, \ell_{S,r}, \ell_{V,r})$. This results in $t + 1$ setup executions of Σ .

For round r , parameters $\ell_{M,r}$, $\ell_{S,r}$, and $\ell_{V,r}$ are upper bounded by $O(n^r)$, $O(n^r)$, and $O(n)$, respectively. To guarantee security against computationally unbounded adversaries, we set $\lambda = \Theta(2^n)$. Combined with the bounds above, in round $t + 1$ (the potential last round) we have an instantiation of Σ with

- signature size $O(n^{t+2})$,
- secret key size $O(n^{2t+3})$,
- (single) verification key size $O(n^{t+2})$,
- key generation time $O(n^{3t+5})$,
- signing time $O(n^{3t+4})$,
- verification time $O(n^{3t+3})$.

We emphasize that when n is fixed (e.g., a constant), all computation and communication complexity bounds become explicit. In such cases, a computationally secure signature scheme with appropriate parameters suffices, provided it satisfies correctness, unforgeability, consistency, and works in linear time in the message size, as shown in [CDL⁺25]. However, in the general setting, an information-theoretic signature scheme is necessary, as parties may run in exponential time. This aspect appears to be overlooked in previous work [ELP25].

B Additional Related Works

In this section, we provide additional related works of BA in general, including (im)possibilities and protocols in various models.

B.1 Feasibilities and Impossibilities

The Byzantine Agreement (BA) problem, also known as the Byzantine General problem, is one of the fundamental problems in distributed computing since it was first introduced in [PSL80, LSP82]. These works showed that, if n denotes the total number of parties and t denotes the maximum number of faulty parties, then $n > 3t$ is necessary for solving BA. One way to circumvent this limitation is to assume an unforgeable signature scheme [GMR88]. Under this assumption, Dolev and Strong [DS83] showed that BA can be solved for any $t < n$.

It is also known that, when n and t are the number of entire parties and the maximum number (threshold) of faulty parties, Fischer and Lynch [FL82] proved that at least $t + 1$ synchronous communication rounds are necessary to solve the BA problem deterministically. On the other hand, Ben-Or [Ben83] and Rabin [Rab83] showed that randomized BAs can circumvent the lower bound. If the network is asynchronous, the celebrated result of Fischer, Lynch, and Paterson [FLP85] proved that solving deterministic BA is impossible even in the presence of only one faulty processor. Garay et al. [GKKO07] pushed the feasibility boundary of randomized BAs up to $t = n/2 + O(1)$. They also established an asymptotically matching lower bound, proving that no randomized BA can terminate in fewer than $2n/(n-t) - 1$ rounds whenever $n - t \geq 2$. Lindell, Lysyanskaya, and Rabin [LLR02] showed that, while authenticated BA protocols can be secure for $t < n$, they require $t < n/3$ to remain secure under parallel or concurrent composition.

B.2 Randomized BAs (with Honest Majority)

One of the first notable works in this regime is Feldman and Micali [FM88], who proposed BA protocols achieving expected constant-round complexity. They considered both synchronous and asynchronous networks, and also both information-theoretic and cryptographic settings. Indeed, randomized BAs can be

classified into those two types: BAs that assume cryptographic primitives—and therefore secure against only computationally bounded adversaries—and BAs that are information-theoretically secure. The line of research [CR93,ADH08,PCR14,CFG+23] studied asynchronous BAs in the information-theoretic setting. In the synchronous model, Goldreich and Petrank [GP90] presented a method to eliminate a failure probability, achieving expected constant-round complexity and worst-case round complexity of $O(t)$ for $t < n/2$. Zam-sky [Zam96] achieved an expected constant-round complexity and the optimal worst-case complexity of $t + 1$ rounds for a resilience of $t < n/8$. Achieving 1) expected constant round complexity, 2) $t + 1$ rounds worst case, and 3) optimal resilience $t < n/3$ altogether is still an open problem.

In the cryptographic setting, Katz and Koo [KK06] and Abraham et al. [ADD⁺19] proposed BA protocols achieving $t < n/2$ and expected constant-round complexity. For the dishonest majority, Fitzi and Nielsen [FN09] proposed randomized protocols with expected round complexity $O(k)$ when $t = n/2 + k$. Chan, Pass, and Shi [CPS20] proposed a sublinear-round BA, and Wan et al. [WXS20] extended the feasibility of expected constant-round Broadcast to tolerate any constant fraction of corruptions.

B.3 Early-Stopping BAs

Dolev, Reischuk, and Strong [DRS90] studied early-stopping protocols to achieve the BA problem, dubbed eventual BA in their work. They proved that, when f is the actual number of faulty parties, deterministic BA protocols should run at least $\min\{t + 1, f + 2\}$ communication rounds in a synchronous network. They also provided a round-optimal early-stopping BA protocol requiring $n > \max\{4t, 2t^2 - 2t + 2\}$. Since then, various researchers have sought to find more efficient early-stopping BA protocols. The first known early-stopping BA achieving both optimal round complexity and resilience (i.e., $t < n/3$) is [BGP92], at the cost of exponential message complexity. The line of research [BGP92,BG93,MW94,ZIP95,GM98,PR04,AD15] has tried to improve and balance between the three quantities: round complexity, message complexity, and resilience. Parvédy and Raynal [PR04] claimed an optimal BA protocol but only tolerating omission faults.¹⁶

The recent breakthrough to achieve early-stopping BA for $t < n$ was made by Loss and Nielsen [LN24] by proposing a new primitive, dubbed ‘polariser’, based on a signature scheme. The polariser allows parties to accuse faulty parties and provide evidence, eventually lets honest parties exclude faulty parties that disturb the termination within $O(f)$ rounds per each such faulty party. This ultimately leads to a Broadcast protocol that achieves $O(\min\{f^2, t\})$ round complexity. Later, Deligios, Klasovita, and Liu-Zhang [DKLZ25] fine-tuned the polariser by carefully dealing with the structure of the communication graph and consequently achieved $O(f)$ -round BA for $t < (1 - \epsilon)n$ for any constant $\epsilon > 0$. Both protocols, however, are far from optimal in terms of round complexity. Another approach to close the gap was taken by Elsheimy, Loss, and Papamanthou [ELP24]. They provided an early-stopping Consensus protocol for $t < n/2$ and any (pre-fixed) positive integer d , and it terminates in $(d+5) \cdot (\lfloor f/d \rfloor + 2) + 2$ rounds. Later, the same authors [ELP25] proposed another protocol by removing the need for a constant d , and achieved $f + 6\lceil\sqrt{f}\rceil + 6$ round complexity. In the same work, the authors also introduced a nearly optimal early-stopping Broadcast protocol for $t < n/2$ that terminates in $\min\{f + 3, t + 1\}$ rounds with exponential message complexity, while the protocol only requires sending a message (but no receiving) in the final round, which turns out to be the major issue of our work.

B.4 BAs in Other Fault Models

Groce et al. [GKTZ12] considered game-theoretic adversaries, i.e., rational adversaries. They showed that perfectly secure BA is possible for $t < n$ when the adversary’s preferences are given. Zikas, Hauser, and Maurer [ZHM09] considered the mixed fault model with Byzantine omission faults. In this line of research, Eldefrawy, Loss, and Terner [ELT22] showed a lower bound in this model, and Loss and Stern [LS23] showed a matching protocol.

¹⁶ To be more precise, their protocol achieves a stronger property than the usual agreement property—uniform agreement, which requires the agreement among all parties including faulty parties. It is not hard to see that the property requires $f + 2(r)$, so the protocol is indeed tight for uniform Consensus.

Table of Contents

On the Round Complexity of Early Stopping	1
<i>Wonseok Choi, Wookjin Choi, Xiangyu Liu, and Vassilis Zikas</i>	
1 Introduction	1
1.1 Overview of Our Contributions	3
2 Technical Overview	6
2.1 Receive-Round $f + 2$ Lower Bound for Byzantine Faults	7
2.2 Send-Round $f + 3$ Lower Bound for Broadcast with Dishonest Majority	7
2.3 $f + 2(s)$ Broadcast Protocol Tolerating Omission Faults	8
2.4 Revisiting BG93	8
2.5 Early-Stopping Broadcast for $t < n/2$	9
3 Preliminaries	9
3.1 Problem Setting and Assumptions	10
3.2 Exponential Information Gathering (EIG)	11
3.3 Notation Conventions for BA Protocols	14
4 A Refined Model to Count Rounds	15
5 Revisiting the Lower Bounds	16
5.1 $f + 2(r)$ Lower Bound for Byzantine Faults	16
5.2 $f + 3(s)$ Lower Bound for Broadcast with Dishonest Majority	20
6 An Optimal Broadcast Protocol Against Omission Faults	22
7 Revisiting Cloture Votes	24
7.1 Review of the Cloture Votes Protocol	25
7.2 Resolve Rules and Fault Detection	28
7.3 Revised Cloture Votes: An $f + 2(r)$ Consensus Protocol	29
8 $f + 2(r)$ Early-Stopping Broadcast Protocol for $t < n/2$	38
8.1 Additional Preliminaries	39
8.2 Description of Rules	40
8.3 Protocol Structure	43
8.4 Proof of Theorem 7	45
9 Conclusion	52
A Information-Theoretic Signatures	56
B Additional Related Works	57
B.1 Feasibilities and Impossibilities	57
B.2 Randomized BAs (with Honest Majority)	57
B.3 Early-Stopping BAs	58
B.4 BAs in Other Fault Models	58