Towards Optimal Early Stopping Agreement Protocols

Fatima Elsheimy[1], Julian Loss[2], and Charalampos Papamanthou[1]

[1] Yale University
[2] CISPA Helmholtz Center for Information Security

Abstract. Early stopping agreement protocols guarantee termination based on the actual number of malicious parties, $f \leq t$, encountered during execution, rather than assuming the worst-case scenario of $t < n$ many corruptions. The lower bound on the round complexity for such protocols is known to be $\min\{f+2, t+1\}$ many rounds. In this work, we substantially improve the state of the art for cryptographic early stopping protocols in the honest majority setting where $t < n/2$. In this scenario, the best known protocol due to Elsheimy, Loss, and Papamanthou (ASIACRYPT '24) achieves a round complexity of $(1+\epsilon)\cdot f$ for some constant $0 < \epsilon < 1$. We begin by introducing an improvement to the Elsheimy et al. approach which can be used to obtain the first polynomial-time early stopping agreement protocols in the $t < n/2$ corruption regime which achieve a complexity of $f + O(\sqrt{f})$. We then instantiate our generic framework with refined components to obtain a very concretely efficient protocol with a round complexity of only $f + 6\lceil\sqrt{f}\rceil + 6$ and polynomial communication complexity. Finally, we show that it is possible to come within one round of the optimal round complexity by presenting a broadcast protocol based on the exponential information gathering approach, which achieves a round complexity of $\min\{f+3, t+1\}$, albeit with exponential communication complexity.

1 Introduction

Byzantine Agreement (BA) is a fundamental problem, first introduced in the seminal work of Lamport, Shostak, and Pease in the 1980s [1]. In the BA problem, n parties, each starting with a value from $\{0, 1\}$, aim to reach agreement on a single value despite the presence of up to $t < n/2$ Byzantine (malicious) parties. The protocol must satisfy two key properties: *Agreement*, which ensures that all honest parties decide on the same value, and *Validity*, which guarantees that if all honest parties start with the same value, they must agree on that value. A common metric to measure the efficiency of a BA protocol is its *round complexity*, i.e., the number of synchronous communication rounds required to reach termination for all honest parties. A fundamental result by Dolev et al. [2] establishes a lower bound on round complexity for the problem BA. Letting t denote an upper bound on the number of corrupted parties that a BA protocol Π can tolerate, their bound states that, for any execution where $f \leq t$ parties are *actually corrupted* $\min\{f+2, t+1\}$ rounds are necessary for Π to terminate *in the worst case*. For executions where $f \ll t$, this lower bound leaves open the exciting possibility of an *early stopping* BA protocol Π that terminates within a short amount of $O(f) = o(t)$ many rounds.

In the information-theoretic setting with $t < n/3$, Abraham and Dolev [3] gave an optimal early stopping protocol with polynomial communication complexity and round complexity $\min\{f+2, t+1\}$. By comparison, in the authenticated setting with $t < n/2$ malicious corruptions, no optimal solution has yet been proposed. The earliest attempt was by Perry and Toueg in 1985 [4], which achieved a sub-optimal round complexity of $2 \cdot \min\{f+2, t+1\}$. More recently, progress has been made by Elsheimy, Loss, and Papamanthou (ELP24) [5], who showed a protocol with round complexity $(1+\epsilon) \cdot f$, where $0 < \epsilon < 1$ is a constant. An important open question is whether it is possible to further narrow the gap toward the optimal round complexity. In this work, we improve significantly upon both of these works for the setting with $t < n/2$ corrupted parties as follows.

– Building on the ideas of Elsheimy et al. we show a generic approach of achieving early stopping BA in $f + O(\sqrt{f})$ many rounds with polynomial complexity. In a second step, we instantiate our generic compiler with new and ameliorated components, which leads to an early stopping protocol with a round complexity of $f + 6\sqrt{f} + 6$.
– While our polynomial-communication protocol achieves, for the first time, early stopping round complexity with a leading coefficient of 1, it leaves open a vexing gap to the aforementioned $\min\{f+2, t+1\}$ lower bound. By relying on the exponential information gathering (EIG) paradigm, we show that it is indeed possible to approach this bound up to one round. However, finding a trivial solution for an early stopping protocol, even with exponential communication, is not straightforward. Our protocol, while incurring exponential complexity, serves as a feasibility result and an important starting point for future endeavors in this direction.

In the following sections, we describe our results in more detail.

1.1 Early Stopping with Polynomial Complexity and $O(\sqrt{f})$ Overhead

We start by giving a recap and overview of the approach of Elsheimy et al. [5] which will be instructive to explain our contribution.

Recap: The Approach of Elsheimy et al. The early stopping BA in ELP24 runs in iterations. Each iteration runs for $d + 5$ rounds, where d is a predefined, but arbitrarily tunable parameter. Their protocol guarantees that at the end of each iteration, honest parties either agree on a common output or that they collectively detect an additional d malicious parties. Once no malicious parties are left to detect, the honest parties therefore agree and eventually are able to terminate from the protocol. It is easy to see that this happens at the latest after

$\lfloor f/d \rfloor + O(1) = f/d + O(1)$ iterations, which implies a total round complexity of $(d+5)\cdot(f/d+O(1)) = f + 5f/d + O(d)$ rounds, which can be simplified to $(1+\epsilon)\cdot f$, whenever $f = \Omega(1)$ and d is a constant.
Achieving the optimal early stopping round complexity of $\min\{f+2, t+1\}$ requires a protocol to eliminate one malicious party per round on average for each iteration in which the protocol *does not terminate*. As explained above, however, the ELP24 protocol eliminates one party only every $1+5/d$ rounds on average for every round where the adversary is able to prevent the termination of honest parties. This rate of detection is inherited from the protocol's internal components, which we will explain in more detail below. An intriguing question is therefore whether it is helpful to increase the parameter d and improve this rate. While this does indeed help to bring down the necessary number of iterations $\lfloor f/d \rfloor$ to detect f parties, the round complexity of each *single* iteration grows. Thus, prior to the final iteration of the protocol, the remaining number of undetected malicious parties may be very small compared to the iteration's length. In this case, the final iteration wastes many rounds during which no further malicious parties can be detected. For example, with $f = 0$ and $d = \log n$, ELP24 already performs worse than Perry and Toueg [4] for $n \geq 4$.

Iterations of Dynamic Length. As explained above, setting d as a fixed parameter in the approach of ELP24 seems to lead to an inherent dilemma: choosing d too small leads to a poor average rate of detected malicious parties per round, whereas a large choice of d can incur a steep penalty toward the end of the protocol when only few malicious parties remain undetected. To overcome this limitation of the ELP24 approach, we propose to set d *dynamically* instead of as a fixed parameter. However, choosing the correct rule for setting d in each iteration turns out to be tricky. To see why, it is instructive to begin with a simple strawman approach where, initially, $d = 1$ and is then doubled during every subsequent iteration. Note that this means that the duration of iteration k will be $2^{k-1}+5$ rounds when directly using the protocol components of ELP24. At first glance, this approach indeed leads to a much lower number of only about $\log_2(f)$ iterations until all honest parties have been detected. Unfortunately, however it drastically overshoots in a scenario where at the end of the penultimate iteration k of the protocol, all malicious parties have been detected, but honest parties are still in disagreement. Note that due to the doubling rule, we have that $k \approx \log_2(f) - 1$. Thus, the final iteration $k+1 = \log_2(f)$ of the protocol will run for $2^k + 5 = f + 5$ rounds without detecting even a single malicious party more.

A Balanced Approach. From the above, it is evident that a more nuanced rule for dynamically setting d is needed. Ideally, we would like to find a way to balance the number of iterations against the duration of the final iteration as much as possible. As we will now explain, incrementing d by 1 for every subsequent iterations achieves exactly this.[3] To see why this is true, let us first see how many iterations it will require the ELP24 approach to eliminate all malicious parties.

[3] As it turns out, incrementig d by 2 every iteration yields a slightly better round complexity, but this is not relevant for the ensuing discussion.

To make the dependency of d on the iteration number explicit, we will denote the choice of d associated with iteration k as d_k below. In line with the approach of incrementing by one in each iteration, we get $d_k = k$ for each iteration and a length of $k + 5$ for each iteration k. Thus, we can see that in $\lceil\sqrt{2f}\rceil$ iterations, the ELP24 approach eliminates $\sum_{k=1}^{\lceil\sqrt{2f}\rceil} d_k = \sum_{k=1}^{\lceil\sqrt{2f}\rceil} k \geq f$ many corrupted parties from the protocol. On the other hand, the length of iteration $\lceil\sqrt{2f}\rceil + 1$ is only $(\lceil\sqrt{2f}\rceil + 1) + 5$. As only one further iteration is required in ELP24 after eliminating the last malicious party, the total number of rounds in the protocol can be computed as $\sum_{k=1}^{\lceil\sqrt{2f}\rceil+1}(d_k + 5) = \sum_{k=1}^{\lceil\sqrt{2f}\rceil}(k + 5) = f + O(\sqrt{f})$.

1.2 Ameliorating the Round Complexity

In the previous section, we have described our approach for designing an early stopping protocol with $f+O(\sqrt{f})$ round complexity and polynomial computation complexity. While this settles the question of obtaining a leading coefficient of 1 in the round complexity of early stopping BA for the $t < n/2$ setting, the hidden constant inside the $O(\sqrt{f})$ turns out to be quite large, i.e, ≈ 10. Following the initial motivation of ELP24 of reducing the leading coefficient from 2 to $(1+\epsilon)$, we show several subtle modifications to the ELP24 approach as well as its internal components. Taken together, these improvements result in a far more round-efficient protocol with a round complexity of only $f + 6\lceil\sqrt{f}\rceil + 6$. We discuss these optimizations in more detail below.

Reducing the Number of Iterations in ELP24. Before diving into the internal components of ELP24, we first show a way of reducing the number of iterations needed for terminating safely in ELP24 by one. While this may, at first, sound like an incremental improvement, recall that the final iteration k in our above approach is by far the most round-expensive one, as we have incremented d_k to size about $\sqrt{2f}$ at this point. Thus, avoiding it saves about that many rounds.

As previously discussed, once all malicious parties have been identified and eliminated in iteration k, the honest parties in ELP24 will be in agreement by the end of iteration $k+1$. Moreover, they will remain in agreement for all subsequent iterations of the protocol. To not have to keep running the protocol indefinitely, however, ELP24 uses a detection mechanism that indicates agreement for all honest parties at most one iteration after they have first reached it. In the above scenario, this implies that agreement is detected at the end of iteration $k + 2$ by all honest parties at the latest. Any honest party that detects agreement sends a signed termination message to all parties. Once $t + 1$ of these messages have been collected for a single bit b, an honest party forwards these messages and terminates the protocol. Since all honest parties sign only the value they detected agreement on, this rule yields a safe way of breaking from the infinite sequence of iterations within roughly two iterations after detecting the last malicious party.

We optimize this termination strategy as follows. Recall that at the end of any iteration k where parties *do not* agree, d_k malicious parties are commonly detected. Conversely, if an honest party P_i observes that fewer than d_k parties have been commonly detected, it can conclude that honest parties are in agreement at the end of iteration k. Thus, from the above discussion, P_i knows that the agreement detection mechanism will be triggered for all honest parties by the end of iteration $k+1$. Therefore, we can avoid running the protocol for iteration $k+2$ entirely.

Recap: Graded Consensus from Correct-Or-Detect Broadcast. The ELP24 protocol is constructed using several components, with its core building block being the Correct-Or-Detect (CoD) Broadcast protocol introduced by Fitzi and Nielsen [6]. The CoD protocol operates in two modes: "Correct" (denoted as C) and "Detect" (denoted as D). If an honest party terminates in mode C, all parties agree on the sender's value. If an honest parties terminates in mode D, all honest parties detect a common set of at least d corrupted parties. However, there is no consensus among the parties in which mode they terminate or who the common parties are.

ELP24 now show how to build a *graded consensus* protocol as follows: each party P_i uses CoD to broadcast its input. It then observes the outputs of the CoD instances of all other parties. If P_i observes $t+1$ instances terminate on value b in mode C, it outputs b and grade $g_i = 1$. On the other hand, for grade $g_i = 0$, P_i simply takes the majority bit over all instances of eligible parties (regardless of what mode they terminate in). Recall that honest parties agree on the output bit b of any instance in which at least one honest party P_i terminated in mode C. Therefore, it immediately follows that if an honest party P_i observes $t+1$ C-instances with bit b and outputs b with grade $g_i = 1$, all other honest parties also output bit b, albeit with grade 0. On the other hand, if parties *disagree* on the output bit b, this implies that at least one of the CoD instances must have terminated in mode D for all of the honest parties. Thus, d new malicious parties can be commonly identified and excluded from further participating in the overall protocol, as we describe below. The above graded consensus protocol runs in $d+4$ rounds (where the round complexity is directly inherited from the Fitzi-Nielsen CoD construction).

Recap: From Graded Consensus to BA. Finally, to achieve Byzantine agreement, ELP24 runs the above graded consensus protocol in iterations as already described in the sections above. In each iteration k, parties use as input the output bit b from the previous iteration $k-1$ and update their lists of faulty parties based on the detections from the previous iteration.

To permanently exclude commonly detected malicious parties from further participation in the protocol, ELP24 again relies on the ideas of Fitzi and Nielsen. At the beginning of each iteration k, honest parties run protocol Π^k_{PoP} (Fig. 2) before invoking CoD. Π^k_{PoP} consists of a single round where all currently undetected parties obtain a "proof of participation" (PoP) which effectively acts as a participation token for the current iteration k. Any commonly detected party, on the other hand, will not obtain a PoP and is thus banned from participating in further iterations. This ensures that honest parties can produce valid PoPs, enabling their continued participation, while detected corrupted parties cannot, effectively excluding them from future rounds. Π^k_{PoP} turns out to be very simple:

a party P_i sends to P_j a signature on the tuple (j, k) for any party P_j which it *has not* detected. P_j collects $t+1$ such signatures as a proof of participation for iteration k. Since parties are required to attach their PoP for iteration k to all protocol messages they send in iteration k, this guarantees that commonly detected parties are banned from participating in subsequent iterations.
When a party P_i observes grade $g_i = 1$ for bit b in iteration k, it is confident that agreement has been reached, since all other honest parties have the same output, albeit with grade 0. Now, all honest parties use bit b in the next iteration $k+1$ and it immediately follows that all honest parties will output bit b with grade 1 in iteration $k+1$ and all future iterations. To break from the iteration loop, a party that has observed grade 1 on an output bit b sends a signed termination message for bit b and continues to run in the next iteration. Upon receiving $t+1$ termination messages for the same value, it forwards these messages in the next round and terminates. In the above scenario, this ensures that all honest parties terminate by the end of iteration $k+1$.
ELP24 uses CoD [6] as a black-box, which detects d malicious parties in $d+4$ rounds. Combining it with Π_{PoP} adds an extra round, leading to a total of $d+5$ rounds to detect and eliminate d malicious parties.

Graded Consensus with Improved Detection Rate. To further improve the round complexity of our protocol, we revisit the CoD protocol of Fitzi and Nielsen. Their protocol, which remains secure against any number of $t < n$ malicious faults and runs in $d+4$ round to detect d malicious parties, is a variation of the classic Dolev-Strong protocol. The key idea of their protocol is for a sender to send a signature on 1 in the first step on input 1 and send no message throughout the entire protocol on input 0. In any round r such that $d+1 \geq r \geq 2$, a party P_i that is not the sender will output 1 in mode C from the protocol if it receives a nested chain of signatures on 1 from r distinct parties, including the sender. Upon receiving such a chain in round in round r for the first time, P_i extends the chain to length $r+1$ by adding its own signature to it and forwarding it to all parties so that they can accept it by round $r+1$. Moreover, P_i adds the first $r-1$ parties to its list F_i of detected malicious parties. This rule of detection is sound since any of these parties should have sent P_i the signature chain in an earlier round by the protocol's instructions, but failed to do so. The final three rounds of the protocol are somewhat more subtle and we do not explain them in detail here. At an abstract level, they ensure that parties agree on the output bit in case an honest party outputs in mode C whereas all honest parties detect d common malicious parties in case an honest party outputs in mode d. One important aspect (and source of inefficiency) of the protocol lies with the fact that parties must wait until the end of round $d+4$ to determine that the sender (honestly) sent 0 by abstaining from sending anything. In ELP24, a further round is lost without detecting additional malicious parties by running protocol Π^k_{PoP} to issue PoPs for participating in the next iteration k of the protocol. Our graded consensus protocol significantly improves on these two sources of inefficiency. First, we leverage the honest majority setting available to our protocol to detect early that a sender is not sending anything. This helps our protocol to waste fewer rounds toward the end and detect d parties in only $d+2$ rounds. At the same time, we show that it is possible to run Π^k_{PoP} for iteration k *in parallel* with the first round of the graded consensus for this iteration without harming security, while retaining a detection rate of d parties per $d+2$ rounds.

1.3 A Broadcast Protocol with Nearly Optimal Round Complexity

The final part of our paper presents a broadcast protocol with a round complexity of $min(f+3, t+1)$, based on the Exponential Information Gathering (EIG) paradigm. This work follows in the footsteps of the foundational early-stopping agreement protocols proposed in [7] for the $t < n/4$ setting and [3] for the $t < n/3$ setting, both of which achieve an optimal round complexity of $min(f+2, t+1)$ for the Byzantine agreement problem. We proceed by providing a general overview of their solutions and then outlining the main idea behind our protocol.

Overview of [7] and [3]. Both protocols use a similar solution framework that starts with the EIG model [1]. The EIG framework revolves around constructing a tree-like structure to store and send information among parties. Each party begins by constructing its own EIG tree, where the root node represents the value to be eventually determined and output by the parties. The children of the root node store the first messages received from other parties, representing the parties' initial inputs, including the input of the party itself. As the protocol progresses over multiple rounds, newly received messages are stored as leaf nodes in the tree and will be forwarded in the next round. This process goes on for $t+1$ rounds. Each party then computes the final value of the root node by applying a deterministic *resolving* function, typically the majority function in the $t < n/3$ and $t < n/4$ settings, recursively from leaves up to the root. That is a node's final value is v if majority of its children has final value v. Ultimately, each party outputs the computed value of the root node. A key limitation of solutions based on the EIG framework is their inherently exponential communication complexity, as messages exchanged during the $t+1$ rounds grow rapidly in every round. To achieve early stopping round complexity, these protocols introduce rules that allow parties to prune certain branches of the EIG tree and stop reporting information on those branches. These rules take advantage of the fact that parties can often predict the final values of some nodes well before the protocol completes, and thus, can stop reporting information on the tree in a number of rounds that's proportional to the exact number of malicious parties in the system, $f \leq t$. In cases where $t < n/3$ or $t < n/4$, early stopping highly depends on the majority function as every node at any level in the tree contains an honest majority. However, for the more challenging $t < n/2$ setting, this property no longer holds, complicating the early stopping rules. Finally, to get polynomial communication complexity, both [7] and [3] incorporate a final step called cloture voting. The core idea is that their early-stopping EIG protocol ensures termination within a constant number of rounds, c, if all honest parties begin with the same initial value, achieving validity. If the protocol does not terminate within c rounds, ensuring validity is no longer necessary, and parties can terminate quickly with a default value. To achieve this, a new EIG protocol is initiated in each subsequent round, where parties effectively "debate" whether to continue or terminate. Of course, additional coordination is required between these different EIG protocols.

Our Approach and Restrictions. In our early-stopping protocol, we designed new rules that move away from relying on the majority value of a node's children. In the $t < n/2$ setting, each honest node is guaranteed to have at least $n - f - |\sigma|$ honest children, where $|\sigma|$ represents the depth of the node. However, as a node appears deeper in the tree, it may have fewer than t honest children. As a result, the deterministic function used at the end of the protocol to compute the final value for each node cannot depend on a simple majority, requiring a different approach. Our protocol follows a 1-biased strategy: a party sends a message if its value is 1 but remains silent if its initial value is 0. Consequently, the early stopping rules are also 1-biased, requiring fewer children of a node σ with value 1 to predict that σ will take the value 1, compared to predicting 0. To determine the final output, we similarly apply the resolving function from the leaves up to the root. This function assigns a node the value 1 if it has at least $t+1-|\sigma|$ children with value 1; otherwise, it assigns 0. To address the exponential communication complexity, the cloture voting idea does not directly apply, as our protocol does not guarantee termination in a constant number of rounds when validity holds. Instead, a different method must be integrated into our EIG protocol to mitigate such communication.

1.4 Related Work

The study of Byzantine agreement has a rich history, beginning with the foundational work of Shostak, Pease, and Lamport [1]. One of the earliest milestones in this area was achieved by Dolev and Strong [8], who proved that any protocol designed to tolerate $t < n$ malicious parties must operate for at least $t+1$ rounds. This lower bound was later improved upon by Dolev et al. [9], who established that when the number of actual corruptions, f, is considerably smaller than t, the required rounds reduce to $\min(f + 2, t + 1)$. This result sparked significant interest in developing early stopping protocols, which aim to terminate more efficiently when fewer faults occur.

The first early stopping protocol in the information-theoretic setting with optimal resilience of $t < n/3$ was introduced by Berman et al. [10], based on the EIG paradigm. As expected, this protocol incurred an exponential communication overhead, similar to our EIG-based protocol. A follow-up work by Garay and Moses tackled this issue and proposed a protocol with polynomial communication complexity [11,12], relying on the cloture voting technique discussed earlier in the technical overview. However, their protocol featured a slightly suboptimal early stopping round complexity of $\min(f + 5, t + 1)$. A major breakthrough was achieved by Abraham and Dolev [3], who introduced the first protocol that attained polynomial communication, optimal resilience, and an optimal round complexity of $\min(f + 2, t + 1)$.

While early stopping protocols have been extensively studied in the information-theoretic setting, progress in the authenticated setting with optimal resilience of $t < n/2$ has been comparatively limited. Notable contributions include the protocol by Perry and Toueg [4], which achieved polynomial communication and a round complexity of $\min(2f + 4, 2t + 2)$, as well as the recent protocol by Elsheimy, Loss, and Papamanthou (ELP24) [5], which achieves a round complexity of $(d + 5) \cdot (\lfloor f/d \rfloor + 2) + 2$, where d is a predetermined constant.

Other works [13,14,15] have explored early stopping protocols under weaker adversarial models, such as omission or crash failures. Additionally, recent studies [16,17] have addressed the early stopping problem in the dishonest majority setting ($t < (1 - \epsilon)n$). The work in [16] provides a protocol for $\epsilon = 0$ with a round complexity of $O(\mathsf{min}\{f^2, t\})$, whereas [17] presents a solution for $\epsilon > 0$ with a round complexity of $O(f, t^2)$, where the leading coefficient of f is at least 4 for $\epsilon = 1/2$.

For randomized protocols, it has been established that they can achieve an expected constant number of rounds in both the information-theoretic setting [18] and the authenticated setting [19,20,21,22]. Additionally, they can achieve termination in a round complexity $O(\lambda)$ which is independent of the number of parties. However, the failure probability of such protocols is at least $O(\lambda^{-\lambda})$ [23]. This dependency complicates direct comparisons between randomized and early stopping protocols, as the latter may require significantly fewer rounds when the number of corruptions, f, is small. For instance, consider the setting with $t < n/2$, $f = 4$ and $\lambda = 128$. The best known randomized protocol [24] in this setting always requires $3/2 \cdot 128 = 192$ rounds to terminate with probability $1-2^{-\lambda}$ for all honest parties. By comparison, our polynomial early-stopping protocol runs for only $4+6 \cdot \lceil\sqrt{4}\rceil + 6 = 4 + 12 + 6 = 22$ rounds. For $t < n/2 \cdot (1-\epsilon)$, the randomized protoccol of Ghinea et al. [25] achieves tight worst-case running times with respect to the aforementioned optimal failure probability $O(\lambda^{-\lambda})$. However, when $t = 0.49n$, the running time of their protocol will only improve over that of [24] when the probability of failure is required to be less than approximately 2^{-200}. At this point, the protocol already has to run for nearly 300 rounds.

1.5 Paper Organization

Section 2 provides the necessary definitions and describes our network model. In Section 3.1, we introduce a generalized framework for the ELP24 Byzantine agreement protocol, incorporating the dynamic parameter d improvement and the modified termination rule. We then prove its correctness and analyze its round complexity. Section 3.2 instantiates this general framework using our newly proposed d-Detecting Graded Broadcast construction, with detailed correctness proofs deferred to Appendix A. Section 4 presents our Broadcast protocol, which is based on the EIG paradigm and integrates our early stopping rules, with correctness proofs provided in Appendix B.

2 Preliminaries

We begin by introducing the model as well as basic definitions.

Network and Setup Assumptions. We assume a a fully connected network of pairwise, authenticated channels between n parties $\{P_1, \ldots, P_n\} = \mathcal{P}$. We consider the *synchronous network model* where all parties have access to a synchronized clock and there is a known upper bound Δ on the message delays of honest parties. This allows parties to run protocols in a round-by-round fashion where rounds are of length Δ and any message that is sent by an honest party at the beginning of a round are delivered by the end of that round to all honest parties. Parties are assumed to have established a public key infrastructure (PKI) of a digital signature scheme that provides an efficient signing routine Sign and an efficient verification routine Verify. Every party P_i is associated with a public key pk_i that is known to all parties and where (only) P_i knows the corresponding secret key sk_i. This allows a party P_i to create a signature $\mathsf{sig}(m)$ on message m using its secret key sk_i via $\mathsf{sig}(m) := \mathsf{Sign}(\mathsf{sk}_i, m)$. $\mathsf{sig}(m)$ can then be efficiently verified by running $\mathsf{Verify}(\mathsf{pk}_i, \mathsf{sig}(m), m)$. We refer to a signature $\mathsf{sig}(m)$ as *valid* if $\mathsf{Verify}(\mathsf{pk}_i, \mathsf{sig}(m), m) = 1$. For ease of notation, we use the abbreviated notation $\langle m \rangle_i$ to refer to tuples $(m, \mathsf{sign}(m, \mathsf{sk}_i))$ throughout the paper.

Adversary Model. We consider an adaptive, computationally-bounded Byzantine adversary that can corrupt up to $t < n/2$ parties at any point of a protocol execution. A corrupt (or malicious) party P_i is under full control of the adversary and may deviate arbitrarily from the protocol. In particular, the adversary learns P_i's signing key sk_i, which allows it to sign messages on P_i's behalf. In addition, we allow the adversary to delete (or replace with its own) any undelivered messages of a newly corrupted party P_i that P_i sent while it was still honest. We denote the set of corrupted parties as $\mathcal{C}$ and the set of uncorrupted (or honest) parties as $\mathcal{H}$.

Next, we give formal definitions to the Byzantine Agreement problem, and other key subroutines.

Definition 1 (Byzantine Agreement). *Let Π be a protocol executed among parties $P_1, ..., P_n$, where each party P_i holds an input $v_i \in \{0, 1\}$ and outputs a value $y_i \in \{0, 1\}$ upon terminating. A protocol Π achieves* Byzantine Agreement*, if the following properties hold whenever at most t parties are corrupted.*

- *Validity: If every honest party P_i inputs $v_i = v$, then all honest parties output $y_i = v$;*
- *Consistency: All honest parties output the same value v.*
- *Termination: Every honest party terminates.*

We also define the sender centric variant of this problem, which is known as *Byzantine broadcast.*

Definition 2 (Byzantine Broadcast). *Let Π be a protocol executed among parties $P_1, ..., P_n$, where a designated sender P_s holds an input $v \in \{0,1\}$ and all parties P_i output a value $y_i \in \{0,1\}$ upon terminating. A protocol Π achieves* Byzantine Broadcast*, if the following properties hold whenever at most t parties are corrupted.*

- *Validity: If the sender is honest and inputs v, then all honest parties output $y_i = v$;*
- *Consistency: All honest parties output the same value v.*
- *Termination: Every honest party terminates.*

Next, we define the d-detecting graded broadcast. A designated sender starts with an initial value, and each honest party P_i inputs a faulty list, F_i. Malicious parties that appear in the initial faulty lists of all honest parties are excluded from the protocol. Additionally, any malicious parties identified during execution are added to each party's updated faulty list F_i^* which is part of P_i's output. It ensures that either (1) all honest parties agree on the sender's value or (2) that d additional malicious parties are added to the intersection of honest parties' updated output list F_i^*, i.e., are *commonly detected.* Here, additional means that none of these d parties where in the intersection of the honest parties' initial input lists F_i, i.e., they were newly detected during the protocol's execution. Since parties do not know which scenario they are in, they additionally output a grade indicating their confidence in their output. If any party P_i outputs the high grade $g_i = 1$, then agreement is implied for all other honest parties. However, another honest party P_j might not be aware of being in agreement, as it may have output $g_j = 0$.

Definition 3 (d-Detecting Graded Broadcast). *Let Π_d be a protocol parametrized by an integer d where all parties input a list $F_i \subset \mathcal{P}$ and the designated sender $P_s \in \mathcal{P}$ additionally inputs $v_s \in \{0,1\}$. Each party P_i outputs a value $y_i \in \{0,1\}$, a grade $g_i \in \{0,1\}$, and an updated list F_i^* s.t. $F_i \subset F_i^* \subset \mathcal{P}$. Π_d achieves* d-Detecting Graded Broadcast *if the following properties hold whenever at most t parties are corrupted and for all honest parties P_i, F_i contains only corrupted parties:*

- *Graded Validity: In all compliant executions, if the sender P_s is honest, and the input value $v_s = v$, then for all honest parties P_i, $y_i = v$ and $g_i = 1$.*
- *Graded Consistency: If two honest parties P_i and P_j output $g_i = g_j = 1$, respectively, then $y_i = y_j$*
- *Soundness: Every honest party P_i outputs a faulty list $F_i^\star$ that consists only of corrupted parties.*
- *Termination: Every honest party terminates.*
- *d-Detection: If two honest parties P_i and P_j output $y_i = 1$ and $y_j = 0$, respectively, then an additional d common parties are added to the faulty lists of all honest parties, i.e., $\left|\bigcap_{P_j \in \mathcal{H}} (F_i^\star \setminus F_i)\right| \geq d$.*

The d-Detecting Graded Consensus problem is the consensus counterpart of the previously defined broadcast problem. In this setting, each honest party inputs both an initial value and a faulty list. Similarly, it guarantees that either all honest parties reach agreement on the same value or they collectively identify at least d malicious parties.

Definition 4 (d-Detecting Graded Consensus). *Let Π_d be a protocol parametrized by an integer d where every party P_i inputs a list $F_i \subset \mathcal{P}$ and a bit $v_i \in \{0,1\}$. Each party P_i outputs a value $y_i \in \{0,1\}$, a grade $g_i \in \{0,1\}$, and an updated list F_i^* s.t. $F_i \subset F_i^* \subset \mathcal{P}$. Π_d achieves d-*Detecting Graded Consensus *if the following properties hold whenever at most t parties are corrupted and for all honest parties P_i, F_i contains only corrupted parties:*

- *Graded Validity: If all honest parties P_i have the same input value $v_i = v$, then for all honest parties P_i, $y_i = v$ and $g_i = 1$.*
- *Graded Consistency: If two honest parties P_i and P_j output $g_i = g_j = 1$, respectively, then $y_i = y_j$*
- *Soundness: Every honest party P_i outputs a faulty list $F_i^\star$ that consists only of corrupted parties.*
- *Termination: Every honest party terminates.*
- *d-Detection: If two honest parties P_i and P_j output $y_i = 1$ and $y_j = 0$, respectively, then an additional d common parties are added to the faulty lists of all honest parties, i.e., $\left|\bigcap_{P_j \in \mathcal{H}} (F_i^\star \setminus F_i)\right| \geq d$.*

3 Early-Stopping Byzantine Agreement Protocols

In this section, we introduce a generalized Byzantine Agreement (BA) protocol based on the one proposed in [5], significantly reducing the round complexity from $(d+5)\cdot(\lfloor f/d \rfloor+2)+2$, where d is a fixed constant, to $f+(4+c)\lceil\sqrt{f}\rceil+(4+c)$, where c represents the difference between the number of rounds in an iteration and the maximum number of faulty parties that can be detected per iteration. This improvement achieves a leading coefficient of 1, bringing the round complexity more closely with the optimal early stopping round complexity of $\min(f+2, t+1)$. We begin by introducing an the optimization techniques, which reduces the round complexity, and then proof the correctness and round complexity of the generalized framework.

3.1 Early-Stopping Protocol in $f + O(\sqrt{f})$ rounds

For the generalized framework (Fig.1), we closely follow the Byzantine Agreement protocol described in [5], which runs in consecutive iterations. Each party inputs $v_i \in \{0,1\}$, and outputs $y_i \in \{0,1\}$. In each iteration k, each party invokes a d_k-Detecting Graded Agreement protocol, $\Pi_{d_k\text{-GDA}}$, as a subroutine, where d_k is set in each iteration k as explained below. Parties determine whether to send termination messages based on the subroutine's output; $(y_{\mathsf{GDA}_i}, g_i)$. If a party outputs grade $g_i = 1$ from $\Pi_{d_k\text{-GDA}}$, it sends a termination message. Upon receiving $t+1$ termination messages with the same value, it terminates.

This is different from the approach in [5], which uses a constant value for d across all iterations. Instead, our protocol starts with $d_1 = 1$ and increases d_k by 2 in each subsequent iteration k. This results in a BA protocol, Π_{BA}, with a round complexity of $f + (4+c)\lceil\sqrt{f}\rceil + (4+c)$ as proven in Lemma 7, given that $\Pi^k_{d\text{-}\mathsf{GDA}}$ runs in $d + c$ rounds for all values of d.

Protocol Π_{BA}

- **Input and Initialization:** Let v_i denote P_i's input. P_i sets $\mathsf{halt}_i :=$ false, $y_i, y_{\mathsf{GDA_i}} :=\perp$, $\mathsf{sendTerm} :=\perp$, $F_i := \emptyset$, $d_1 = 1$, and $k = 1$.
- **While** $\mathsf{halt}_i = \mathsf{false}$ **do** (we refer to the following as *iteration* k):
  - P_i runs protocol $\Pi_{d_k\text{-}\mathsf{GDA}}$ with input (v_i, F_i)
  - When $\Pi_{d_k\text{-}\mathsf{GDA}}$ terminates with output $(y_{\mathsf{GDA_i}}, g_i, F_i^\star)$, P_i does as follows:
    - ∗ P_i updates the input of next iteration by setting $v_i := y_{\mathsf{GDA_i}}$ and $F_i = F_i^\star$.
    - ∗ If $g_i = 1$ or $|F_i^\star| - |F_i| < d_k$, P_i sets $\mathsf{sendTerm} = y_{\mathsf{GDA_i}}$ and, if it hasn't been sent already, sends $\langle \mathsf{terminate}, \mathsf{sendTerm}\rangle_i$ to all parties in the next round.
    - ∗ P_i sets $k = k + 1$ and $d_k := 1 + 2(k - 1)$.
- **Output Determination:** Once P_i receives $t+1$ $\langle \mathsf{terminate}, v\rangle_i$ in some round r, it sets $y_i = v$, forwards these messages in round $r + 1$, and continue participating in the protocol for round $r + 1$. At the end of round $r + 1$, it terminates.

Fig. 1. Code of Π_{BA} for party P_i.

In summary, in Π_{BA}, each party to start with an input value $v_i \in \{0, 1\}$ and outputs a value $y_i \in \{0, 1\}$. The protocol runs in iterations, with each iteration k invokes $\Pi_{d_k\text{-}\mathsf{GDA}}$. Each party inputs the tuple (v_i, F_i) to $\Pi_{d_k\text{-}\mathsf{GDA}}$ during each iteration. Based on the grade g_i and the number of newly added faulty parties determined by $\Pi_{d_k\text{-}\mathsf{GDA}}$, each party P_i decides whether termination is safe. If P_i outputs $g_i = 1$, it ensures that all other honest parties P_j output the same value $y_{\mathsf{GDA}_i} = y_{\mathsf{GDA}_j}$, as guaranteed by the graded consistency property of $\Pi_{d_k\text{-}\mathsf{GDA}}$. In this case, P_i sends a termination message $\langle \mathsf{terminate}, y_{\mathsf{GDA}_i}\rangle_i$ to all parties. Similarly, if fewer than d_k new malicious parties are added to P_i's faulty list, P_i ensures that parties have agreement from the d_k-detection property and sends the termination message $\langle \mathsf{terminate}, y_{\mathsf{GDA}_i}\rangle_i$. On the other hand, if P_i outputs $g_i = 0$, it indicates that termination is not yet safe, and additional iterations are needed. In such cases, P_i updates its input for the next iteration using the output value y_{GDA_i} from $\Pi_{d_k\text{-}\mathsf{GDA}}$, setting $v_i = y_{\mathsf{GDA}_i}$ and updating its faulty list as $F_i = F_i^\star$.

Upon receiving $t + 1$ termination messages with the same value, a party forwards these messages in the next round, sets its output to that value, and terminates. Each iteration k consists of $d_k + c$ rounds, corresponding to the round complexity of $\Pi_{d_k\text{-}\mathsf{GDA}}$. Thus, the overall round complexity of Π_{BA} depends on the number of iterations required. We prove in Lemma 2 that the number of iterations is a function of f. Finally, we prove that Π_{BA}, based on $\Pi_{d_k\text{-}\mathsf{GDA}}$, achieves Byzantine agreement as defined in Definition 2 and achieves a total round complexity of $f + (4 + c)\lceil\sqrt{f}\rceil + (4 + c)$.

We proceed by first proving that that honest parties are never included in the faulty lists of other honest parties in any iteration. From this point forward, we assume this lemma holds indefinitely. We then proceed with proving the correctness of Π_{BA} in figure 1.

Lemma 1. *At the beginning of every iteration k of Π_{BA}, the faulty list F_i of every honest party P_i contains only corrupted parties.*

Proof. In the first iteration $k = 1$, the faulty lists of all honest parties are empty, so the lemma holds trivially. For subsequent iterations $k > 1$, each party updates its F_i based on the output of $\Pi_{d_k\text{-}\mathsf{GDA}}$. According to the soundness property of $\Pi_{d_k\text{-}\mathsf{GDA}}$, no honest party P_i is included in the F_j of any other honest party P_j in any of these iterations. Thus the claim follows by a simple induction. □

Lemma 2. *If no honest party has sent a termination message in the first round of iteration k or earlier, then no honest party terminates in any round in iteration k or earlier.*

Proof. According to the protocol, a party sends a termination message only after setting $sendTerm$, which is determined by the output of $\Pi_{d_k\text{-}\mathsf{GDA}}$ in the final round of an iteration. This means a party can only send a termination message in the first round of an iteration, not in any subsequent rounds. By assumption, no honest party sends a termination message in the first round of iteration k or any earlier iteration. As per the protocol, an honest party terminates only if it receives at least $t + 1$ termination messages. However, since there are at most t malicious parties, the adversary cannot produce $t + 1$ valid termination messages without forging signatures. Consequently, no honest party can terminate in iteration k or any earlier iteration.

Lemma 3. *For all $d_k \in \mathbb{N}$ and $d_k \geq 1$, let $\Pi_{d_k\text{-}\mathsf{GDA}}$ be a d_k-Detecting Graded Agreement protocol as per Definition 4. Assume that no honest party has sent a termination message by the beginning of iteration k. If all honest parties P_i have the same input value $v_i = v$ in iteration k, then by the end of iteration k, all honest parties set $g_i = 1$ and terminate with that value within two subsequent rounds.*

Proof. By assumption, no honest party has sent a termination message by the beginning of iteration k, and thus, no honest party has terminated or will terminate by the end of iteration k according to Lemma 2. Let each honest party P_i have $v_i = v$. By Lemma 1, the faulty list F_i of every honest party P_i contains only corrupted parties. Consequently, all honest parties invoke $\Pi_{d_k\text{-}\mathsf{GDA}}$ with the same input value v in the first round of iteration k. From the graded validity of $\Pi^k_{d_k\text{-}\mathsf{GDA}}$, every honest party outputs $y_{\mathsf{GDA_i}} = v$ and $g_i = 1$ by the end of iteration k. Since $g_i = 1$, each honest party sets $y_i := v$ and $sendTerm := v$. In the next round (the first round of iteration $k+1$), each honest party broadcasts $\langle \mathsf{terminate}, v\rangle_i$. By the end of this round, every honest party receives at least $t+1$ termination messages of the form $\langle \mathsf{terminate}, v\rangle_i$. In the subsequent round, they forward these $t+1$ messages and terminate. Since there are at most t malicious parties, the adversary cannot collect $t+1$ termination messages with a conflicting value to cause disagreement. □

Corollary 1. *For all $d \in \mathbb{N}$ and $d \geq 1$, let $\Pi_{d\text{-}\mathsf{GDA}}$ be a d-Detecting Graded Agreement protocol as per Definition 4. Then, Π_{BA} achieves validity per Definition 2*

Proof. Assume all honest parties have the same initial value ($v_i = v$) in iteration 1. According to Lemma 3, where $k = 1$, every honest party P_i terminates with $y_i = v$ in the 2nd round of iteration 2. □

Lemma 4. *Let the termination message $\langle \mathsf{terminate}, v\rangle_j$ be the first termination message sent by some honest party in iteration k, then all honest parties send $\langle \mathsf{terminate}, v\rangle$ by the first round of iteration $k+1$ at the latest. Furthermore, no honest party P_i sends a termination message of the form $\langle \mathsf{terminate}, v'\rangle_i$ with $v' \neq v$ in iterations k or $k+1$.*

Proof. According to the protocol, a party sends a termination message only after setting $sendTerm$, which is determined by the output of $\Pi_{d_k\text{-}\mathsf{GDA}}$ in the final round of an iteration. This means a party can only send a termination message in the first round of an iteration. As P_j is the first honest party to send the termination message in iteration k, which occurs in the first round, no honest party has terminated in an earlier iteration according to Lemma 2. According to the protocol, P_j sends a termination message $\langle \mathsf{terminate}, v\rangle_j$ if it either outputs $g_j = 1$ or detects fewer than d_{k-1} malicious parties at the end of iteration $k-1$. By the graded consistency and d_{k-1}-detection properties of $\Pi^k_{d_{k-1}\text{-}\mathsf{GDA}}$, every honest party P_i outputs $y_{\mathsf{GDA}_i} = v$ at the end of iteration $k-1$. As a result, each honest party updates its input value for the next iteration to $v_i := v$. This means that if any other honest party sends a termination message in the first round of iteration k, it will also be on v. During iteration k, if any honest party P_i receives $t+1$ termination messages on v in round $r \leq d_k + c - 2$, it sets $y_i := v$, forwards these messages in the next round, and terminates. All other honest parties receive these $t+1$ termination messages in the next round and terminate by the end of round $r+1 \leq d_k + c - 1$ of iteration k. On the other hand, if no honest party receive such termination messages in round $r \leq d_k + c - 2$, then all honest parties participate in $\Pi_{d_k\text{-}\mathsf{GDA}}$ till the end of round $d_k + c$, as every honest party participates in one more round after receiving these termination messages in round r. From graded validity of $\Pi_{d_k\text{-}\mathsf{GDA}}$, every honest party P_i outputs $y_{\mathsf{GDA}_k} = v$ in iteration k with grade $g_k = 1$. Thus, every honest party sends a termination message $\langle \mathsf{terminate}, v\rangle_i$ in the next round(first round of iteration $k+1$) and not on a different value. □

Lemma 5. *Let P_j be the first honest party to send the termination message $\langle \mathsf{terminate}, v\rangle_j$ in iteration k. Then, all honest parties will terminate with $y_i = v$ no later than the second round of iteration $k+1$.*

Proof. From Lemma 4, every honest party P_i sends a termination message $\langle \mathsf{terminate}, v\rangle_i$ by iteration the first round of iteration $k+1$ at the latest. By the end of round 1, every honest party has received at least $t+1$ termination messages on v, forwards these messages in the subsequent round, and terminates with $y_i = v$. From Lemma 4, no honest party sends a termination message on a conflicting value $v' \neq v$. Since there are at most t malicious parties, the adversary cannot create a conflicting termination certificate for any value $v' \neq v$ and lead honest parties to terminate with value v'. □

Lemma 6. *For all $d \in \mathbb{N}, d \geq 1$, let $\Pi_{d\text{-}\mathsf{GDA}}$ be a d-Detecting Graded Agreement protocol as per Definition 4. Then, Π_{BA} achieves consistency per Definition 2.*

Proof. Assume that P_i is the first honest party to set $y_i := v$ during some round r of an iteration, say iteration k. We prove that every other honest party, denoted as P_k, also outputs v. For P_i to set $y_i := v$, it must have collected at least $t+1$ messages of the form $\langle \mathsf{terminate}, v\rangle_j$ by the end of round $r-1$. Among these $t+1$ messages, at least one must have come from an honest party, say P_j in some iteration $k' \leq k$. From Lemma 5, all honest parties will terminate with $y_i = v$ no later than the second round of iteration $k'+1$. □

Finally, we prove the round complexity of Π_{BA}.

Lemma 7. *Let $c \in \mathbb{N}$ and for all $d \in \mathbb{N}, d \geq 1$, let $\Pi_{d\text{-}\mathsf{GDA}}$ be a d-Detecting Graded Agreement protocol as per definition 4, which runs in $d+c$ rounds. Then, Π_{BA} terminates in $f + (4+c)\lceil\sqrt{f}\rceil + (4+c)$ rounds.*

Proof. If any honest party sends a termination message in iteration k, then from Lemma 5, all honest parties terminate in iteration $k+1$ at the latest. So assume that no honest party sent a termination message so far. In each iteration k, honest parties P_i and P_j either: *(1)* output the same value from protocol $\Pi_{d_k\text{-}\mathsf{GDA}}$, i.e., $y_{\mathsf{GDA_i}} = y_{\mathsf{GDA_j}}$, or *(2)* detect at least d_k additional malicious parties according to the d_k-detection property of $\Pi_{d_k\text{-}\mathsf{GDA}}$. If the former, both P_i and P_j start iteration $k+1$ with the same value and terminate two rounds after iteration $k+1$, according to Lemma 3. Thus, in the worst case, the adversary keeps delaying the agreement on the output of $\Pi_{d_k\text{-}\mathsf{GDA}}$. In the first iteration, the value of d_1 is set to 1 and increases by 2 in each subsequent iteration. Hence, in the k-th iteration, $d_k = 1 + 2(k-1)$. To eliminate f malicious parties, the maximum number of iterations, denoted by s, must satisfy the total sum: $\sum_{k=1}^{s} d_k = \sum_{k=1}^{s} (1 + 2(k-1)) = f$. Thus, $\sum_{k=1}^{s} (1 + 2(k-1)) = s(s+1) - s$, and $s^2 = f$. Therefore, there are at most $\lceil\sqrt{f}\rceil$ iterations required to eliminate all malicious parties. By iteration $\lceil\sqrt{f}\rceil + 1$, every honest party outputs the same value from $\Pi_{d_k\text{-}\mathsf{GDA}}$ due to the d_k-detection property. Furthermore, since they detect less than $d_{\lceil\sqrt{f}\rceil+1}$ parties, each honest party sends a termination message $\langle \mathsf{terminate}, v\rangle_j$ in the first round after iteration $\lceil\sqrt{f}\rceil + 1$ to all other parties. Consequently, each party receives at least $t+1$ termination messages with the same value, it forwards them in the subsequent round and terminates. Note that each iteration k runs for $d_k + c$ rounds. Thus, the overall round complexity is computed as:

$$\begin{aligned}
\mathsf{Complexity} &= \Sigma_{k=1}^{\lceil\sqrt{f}\rceil+1}(1+2(k-1)) + c\cdot(\lceil\sqrt{f}\rceil+1)+2. \\
&= \Sigma_{k=1}^{\lceil\sqrt{f}\rceil+1}1 + \Sigma_{k=1}^{\lceil\sqrt{f}\rceil+1}2(k-1) + (c+1)\lceil\sqrt{f}\rceil + (c+3) \\
&= 2\Sigma_{j=0}^{\lceil\sqrt{f}\rceil} j + (c+1)\lceil\sqrt{f}\rceil + (c+3) \\
&= (\lceil\sqrt{f}\rceil+1)(\lceil\sqrt{f}\rceil) + (c+1)\lceil\sqrt{f}\rceil + (c+3), \ (\text{as } \lceil\sqrt{f}\rceil \le \sqrt{f}+1) \\
&\le (\sqrt{f}+1)^2 + (c+2)\lceil\sqrt{f}\rceil + (c+3) \\
&\le f + (c+4)\lceil\sqrt{f}\rceil + (c+4)
\end{aligned}$$

□

We sum up Corollary 1, and Lemmata 6, and 7 into Theorem 1 as follows:

Theorem 1. *Let $c \in \mathbb{N}$ and for all $d \in \mathbb{N}, d \ge 1$, let $\Pi^k_{d\text{-GDA}}$ be a d-Detecting Graded Agreement protocol as per Definition 4 which runs in $d+c$ rounds. Then, Π_{BA} achieves Byzantine agreement as per Definition 2. Furthermore, Π_{BA} terminates in $f+(c+4)\lceil\sqrt{f}\rceil+(c+4)$ rounds.*

3.2 Early-Stopping Protocol in $f + 6\lceil\sqrt{f}\rceil + 6$ rounds

To achieve $f+6\lceil\sqrt{f}\rceil+6$, we introduce a new $\Pi^k_{d\text{-GDB}}$ protocol that is capable of detecting d malicious parties within $d+2$ rounds. As discussed previously, $\Pi^k_{d\text{-GDB}}$ uses the Proof of Participation protocol (Π^k_{PoP}) [6] as a subroutine. This section an overview of the Π^k_{PoP} protocol as well as relevant definitions, and followed by the construction of the new $\Pi^k_{d\text{-GDB}}$. Subsequently, we use the graded broadcast to design the graded consensus protocol, $\Pi^k_{d\text{-GDA}}$. Finally, we instantiate the BA protocol in Fig. 1 with $\Pi^k_{d\text{-GDA}}$, and prove its round complexity. Note, all our protocols are parameterized by k, which represents the iteration in which they were invoked. This is essential as k must be passed to Π^k_{PoP}, which is invoked by $\Pi^k_{k\text{-GDB}}$.

Next, we define *Proofs of Participation (PoP)*, which a party attaches to its messages to prove its honesty and remain participating in the protocol.

Definition 5 (Proof of Participation). *A* k-proof of participation (k-PoP) PoP^k_i *for a party $P_i \in \mathcal{P}$ and an integer k is a collection of $t+1$ signatures of the form $\langle P_i, k\rangle_{j_l}$, from distinct signers $P_{j_1}, \ldots, P_{j_{t+1}} \in \mathcal{P}$. We say that PoP^k_i is* valid *if for all $l \in [t+1]$, $\langle P_i, k\rangle_{j_l}$ is valid with respect to pk_{j_l}. Further, we assume that the empty string constitutes a valid choice for PoP^1_i, for all $i \in [n]$.*

We modify the signature chain definition from [5] to incorporate PoPs. Each signature in the chain is valid only if it includes a valid PoP and is correctly signed with the party's secret key.

Definition 6 (Signature Chain with PoP). *Let $m \in \{0,1\}^*$, let $i \in [n]$, $k \in \mathbb{N}$ and, for all i, let PoP_i^k be the k-PoP of party P_i as per Definition 5. We write $\langle m \rangle_\sigma$ to denote the nested term $\langle \ldots \langle \langle m, \mathsf{PoP}_{j_1}^k \rangle_{j_1}, \mathsf{PoP}_{j_2}^k \rangle_{j_2}, \ldots, \mathsf{PoP}_{j_l}^k \rangle_{j_l}$, where $j_1, \ldots j_l$ are distinct values in $[n]$, and σ is the ordered sequence of these values, i.e., $\sigma = j_1, \ldots j_l$. We then refer to $\langle m \rangle_\sigma$ as a k-*signature chain of length *l. The expression $\langle m \rangle_\sigma$ is said to be* valid *if, for all l, the signature with respect to pk_{j_l} is valid and the proof of participation PoP_{jl}^k is valid.*

3.2.1 Proof of Participation (Π_{PoP}^k)

The Proof of Participation protocol, Π_{PoP}^k, enables each party to collect a list of honesty proofs for each party $P_i \in \mathcal{P}$, referred to as PoP_i^k, where $k \in [n]$ represents the iteration in which the protocol is invoked. A PoP^k is considered *valid* if it includes $t+1$ valid signatures from distinct parties $P_j \in \mathcal{P}$, in the form $\langle P_i, k \rangle_j$. To generate this list of proofs, each party P_i executes Π_{PoP}^k using its current view of faulty parties, denoted as F_i, as input. In the first round of Π_{PoP}^k, each party broadcasts a PoP^k message, $\langle P_j, k \rangle_i$, for each party P_j not included in F_i. Upon receiving at least $t+1$ such messages for party P_j, a party P_i aggregates them to construct a PoP_j^k proof.

Protocol Π_{PoP}^k

- **Input and Initialization:** Let F_i denote P_i's input.
  P_i sets $\{\mathsf{PoP}_j^k := \perp\}_{j \in [n]}$
- **Round** 1**:**
  - For each party $P_j \notin F_i$, party P_i sends $\langle P_j, k \rangle_i$ to all parties.
- **Output Determination:** If P_i receives valid signatures $\langle P_j, k \rangle_l$ from at least $t+1$ distinct parties, P_i aggregates these messages into PoP_j^k. P_i then outputs the set of PoPs $\mathsf{PoP}^k := \{\mathsf{PoP}_j^k\}_{j \in [n]}$, and terminates.

Fig. 2. Code of Π_{PoP} for party P_i.

Next, we define the two primary properties of Π_{PoP}^k in Lemma 8 and Lemma 9.

Lemma 8. *Assume no honest party P_j is in the faulty list F_i of any other honest party P_i. Then, each honest party P_i outputs a valid proof of participation PoP_j^k for each honest party P_j.*

Proof. There are at most $t < n/2$ malicious parties. Each honest party P_i sends $\langle P_j, k \rangle_i$ for every honest party $P_j \notin F_i$. As per assumption, every honest party P_i will receive at least $t+1$ messages of $\langle P_j, k \rangle_l$ for every honest party P_j. Consequently, every honest party sets PoP_j^k to the aggregation of those received messages. □

Lemma 9. *Assume there exists some party P_j such that $P_j \in F_i$ for all honest parties $P_i \in \mathcal{P}$. Then, no party outputs a valid proof of participation PoP_j^k.*

Proof. There are at most $t < n/2$ malicious parties. No honest party will send $\langle P_j, k\rangle_i$ for party $P_j \in F_i$. Thus, every party can collect at most $t < n/2$ such messages, which are not enough to form PoP_j^k without forging signatures. □

3.2.2 *d*-Detecting Graded Broadcast

To construct the improved $\Pi^k_{d\text{-}\mathsf{GDB}}$ protocol, we depart from the black-box use of the CoD protocol [6], as described in [5]. Instead, we design a new protocol that integrates the Dolev-Strong protocol with a two-round graded broadcast protocol [18]. This approach enables the construction of a d-detecting graded broadcast protocol with guarantees similar to those of the CoD protocol in [5], but achieved in only $d + 2$ rounds while detecting d faulty parties, compared to $d + 5$ rounds and detection of d faulty parties in their approach.

Overall, $\Pi^k_{d\text{-}\mathsf{GDB}}$ (see Fig. 3) is a variant of graded broadcast protocols [18]. However, unlike traditional graded broadcast protocols, $\Pi^k_{d\text{-}\mathsf{GDB}}$ enables honest parties to output not only the broadcast value and a grade but also a list of detected malicious parties. The protocol is parameterized by predefined values d, and k, where d influences the number of protocol rounds; $d+2$ rounds, as well as the number of detected malicious parties, and k indicates the phase in which $\Pi^k_{d\text{-}\mathsf{GDB}}$ is invoked. The round complexity of $\Pi^k_{d\text{-}\mathsf{GDB}}$ differs from typical graded broadcast protocols in the literature, which generally require only 2 or 3 rounds.

In $\Pi^k_{d\text{-}\mathsf{GDB}}$, there is a designated sender with input value $v_s \in \{0, 1\}$. Each party has input faulty list F_i. Each party outputs a value $y_i \in \{0, 1\}$, a grade $g_i \in \{0, 1\}$, and an updated list of identified malicious parties $F_i^\star \subset \mathcal{P}$. In the first round, all parties run protocol Π^k_{PoP} to obtain list of valid k-PoP, including its own. Every party tags along its k-PoP when sending a message and only accepts messages from parties if they tag along their valid k-PoP. Thus, only parties P_i with a valid k-PoP PoP_i^k are allowed to *actively* participate in the d first rounds of the protocol. (As we explain below, they can still participate *passively*, i.e., by sending nothing.). Concurrently, the designated sender P_s sends his value to parties only if it is $v_s = 1$ in the first round; otherwise, he refrains from sending anything. Note, as we run Π^k_{PoP} in the first round, party P_s can not include PoP_s^k along with its message. Thus, we let the receiving parties in the 2nd round to decide on whether to accept the sender's message based on the output of Π^k_{PoP}. Specifically, if $\mathsf{PoP}_s^k \neq \perp$, party P_i constructs a message by concatenating the sender's message, the sender's PoP PoP_s^k, and P_i's PoP PoP_i^k, both generated by Π^k_{PoP}, and forwards it to all other parties in the second round. Thereafter, if an honest party receives a valid chain on 1, it appends its PoP_i^k, signs it and forwards to all parties in the next round. To reduce the round complexity of the CoD protocol [6], we eliminate the need for two additional passive rounds (from $d + 3$ to $d + 4$) previously required for agreement on 0. Instead, we introduce a proactive measure where parties send a $\langle \mathsf{Vote}_0, \mathsf{Nomsg}, \mathsf{PoP}_i^k\rangle$ message in round $d + 1$ if they have not received any chains on 1 prior to this round. In round $d + 1$, each party either sends its vote on 0as $\langle \mathsf{Vote}_0, \mathsf{Nomsg}, \mathsf{PoP}_i^k\rangle$, or on a valid chain for 1 received earlier, by sending $\langle \mathsf{Vote}_1, \langle 1\rangle_\sigma, \mathsf{PoP}_i^k\rangle$. In round $d + 2$, if a party receives a set of $t+1$messages of the form $\langle \mathsf{Vote}_1, \langle 1\rangle_\sigma, \mathsf{PoP}_i^k\rangle$ in the previous round, it forms the set S_1 from these messages and sends it to all parties. Otherwise, if it receives $t+1$messages of the form $\langle \mathsf{Vote}_0, \mathsf{Nomsg}, \mathsf{PoP}_i^k\rangle$ and no vote on 1, it forms the set S_0 from these messages and sends it to all parties. By the end of round $d+2$, each party decides on its grade and output based on the number of sets S_1 and S_0 it receives, with priority given to S_1. Finally, if the designated sender is honest, we ensure that all honest parties output the same value with grade $g=1$. Otherwise, if the outputs differ among parties, we guarantee that all honest parties detect at least the same set of d malicious parties by the end of round $d+1$. Note that parties invoke Π^k_{PoP} in round $d+2$, ensuring that the detected d malicious parties cannot obtain a valid PoP^{k+1} to participate in subsequent rounds. Finally we state the main Lemma for this section and defer the proof to the appendix.

Lemma 10. *$\Pi^k_{d\text{-}\mathsf{GDB}}$ achieves d-Detecting Graded Broadcast as per Definition 3 and terminates in $d+2$ rounds.*

3.2.3 *d*-Detecting Graded Agreement

The protocol $\Pi^k_{d\text{-}\mathsf{GDA}}$ is an extension of the d-Graded Detecting Broadcast (GDB), that achieves Byzantine agreement instead of Byzantine Broadcast. In this protocol, each honest party P_i has an input value v_i and outputs both a final decision value and a list of detected malicious parties. Specifically, $\Pi^k_{d\text{-}\mathsf{GDA}}$ ensures that either all honest parties output the same value y_i, or at least d malicious parties are identified, thereby achieving d-detection.

Each party P_i begins with an input $v_i \in \{0,1\}^\star$, and an initial faulty list F_i. The outputs include a value $y_i \in \{0,1\}^\star$, a grade $g_i \in \{0,1\}$, and an updated list of identified malicious parties $F_i^\star \subset \mathcal{P}$. In the first round, each party P_i invokes $\Pi^k_{d\text{-}\mathsf{GDB}}$ with the input tuple (v_i, F_i). For simplicity, we denote $\Pi^{k,j}_{d+1\text{-}\mathsf{GDB}}$ as the instance of $\Pi^k_{d\text{-}\mathsf{GDB}}$ where P_j is the sender. Each party stores the output $(y_{i,j}, g_i^j, F_i^j)$ from all completed instances of $\Pi^{k,j}_{d\text{-}\mathsf{GDB}}$ for each $P_j \in \mathcal{P}$. To determine the final output y_i and grade g_i, a party P_i considers the results of all $\Pi^{k,j}_{d\text{-}\mathsf{GDB}}$ instances. If there exists a bit $v \in \{0,1\}$ such that for at least $t+1$ instances $\Pi^{k,j}_{d\text{-}\mathsf{GDB}}$, $g_i^j = 1$, P_i outputs $g_i = 1$. To set the final value y_i, P_i outputs the majority value among $y_{i,j}$ from the $\Pi^{k,j}_{d\text{-}\mathsf{GDB}}$ instances. The majority value is guaranteed to align with the grading rules, as proven in Lemma 11. If an honest party P_i outputs $y_i = v$ and $g_i = 1$, then every honest party P_j outputs $y_i = v$.

Finally, similar to $\Pi^k_{d\text{-}\mathsf{GDB}}$, each party P_i forms the updated faulty list $F_i^\star$ by taking the union of all faulty lists output by the $\Pi^{k,j}_{d\text{-}\mathsf{GDB}}$ instances, along with the parties in its initial faulty list F_i. The protocol terminates in $d+2$ rounds as it just runs $\Pi^k_{d\text{-}\mathsf{GDB}}$ without any additional rounds.

In the following lemma, we prove the correctness of $\Pi^k_{d\text{-}\mathsf{GDA}}$ per Definition 4. For space restrictions, the proof of this lemma can be found in the appendix.

Protocol $\Pi^k_{d\text{-}\mathsf{GDB}}$

- **Input and Initialization:** If $P_i = P_s$, let v_i, and F_i denote $P_i's$ input. Otherwise, let F_i denote P_i's input. P_i sets $F^\star_i := \emptyset$, $y_i, v_{temp}, \{\mathsf{PoP}^k_i\}_{i\in[n]} = \bot$ $g_i := 0$.
- **Round** 1 **($P_i = P_s$):**
  - If the sender's initial value is $v_s = 1$, it sets $v_{temp} := \langle 1\rangle_s$, and sends $\langle 1\rangle_s$ to all parties. (Otherwise, it does nothing.)
  - Party P_i runs Π^k_{PoP} on input $F^\star_i$. Let $\{\mathsf{PoP}^k_i\}_{i\in[n]}$ denote the output.
- **Round** 2 **($P_i \neq P_s$):**
  - If P_i received $\langle 1\rangle_s$ from the sender in round 1 and $\mathsf{PoP}^k_s \neq \bot$, P_i sets $v_{temp} := \langle 1\rangle_s$, constructs message $\langle\langle 1\rangle_s, \mathsf{PoP}^k_s, \mathsf{PoP}^k_i\rangle_i$ and sends it to all parties.
- **For any round $\mathsf{r} > 2$ ($P_i \neq P_s$):**
  - If P_i received a valid signature chain of the form $\langle 1\rangle_{\sigma j}$ of length $r-1$ in round $r-1$, it appends to the chain its signature and PoP^k, i.e., it computes $m := \langle\langle 1\rangle_{\sigma j}\mathsf{PoP}^k_i\rangle_i$ and sends m to all parties. Furthermore, it sets $v_{temp} = \langle 1\rangle_{\sigma j}$ if not already set.
  - If P_i received a valid signature chain $\langle 1\rangle_{\sigma j}$ of length r, for $\langle 1\rangle_{\sigma j} = \langle \dots \langle\langle 1, \mathsf{PoP}^k_s\rangle_s \dots \mathsf{PoP}^k_l\rangle_l \mathsf{PoP}^k_j\rangle_j$, P_i adds every party in $P_s, \dots, P_l$ in the signature chain to $F^\star_i$.
- **Round $d+1$:**
  - If $v_{temp} \neq \bot$, it sends $\langle \mathsf{vote}_1, v_{temp}, \mathsf{PoP}^k_i\rangle_i$ to all parties. Otherwise, it sends $\langle \mathsf{vote}_0, \mathsf{Nomsg}, \mathsf{PoP}^k_i\rangle_i$ to all parties.
- **Round $d+2$:**
  - If P_i received a message of the form $\langle \mathsf{vote}_1, \langle 1\rangle_\sigma, \mathsf{PoP}^k_j\rangle_j$ in round $d+1$, where $\langle 1\rangle_\sigma$ is a valid chain, P_i adds $\langle \mathsf{vote}_1, \langle 1\rangle_\sigma, \mathsf{PoP}^k_j\rangle_j$ to set S_1. If $|S_1| \geq t+1$, P_i sends $\langle S_1\rangle_i$ to all parties.
  - If P_i did not receive $\langle \mathsf{vote}_1, \langle 1\rangle_\sigma, \mathsf{PoP}^k_j\rangle_j$ in round $d+1$, such that $\langle 1\rangle_\sigma$ is a valid chain, it adds all received messages in round $d+1$ of the form $\langle \mathsf{vote}_0, \mathsf{Nomsg}, \mathsf{PoP}^k_j\rangle_j$ to set S_0. If $|S_0| \geq t+1$, it sends $\langle S_0\rangle_i$ to all parties.
  - If P_i set $v_{temp} := \langle 1\rangle_{\sigma j}$ in round d, and did not receive at least $t+1$ messages of the form $\langle \mathsf{vote}_1, \langle 1\rangle_\alpha, \mathsf{PoP}^k_l\rangle_l$ in round $d+1$, it adds P_j to $F^\star_i$.
  - P_i sets $F^\star_i = F^\star_i \cup F_i$.
- **Output Determination:**
  - If $P_i = P_s$, then P_i sets $y_i := v_s$, $g_i := 1$, and terminates.
  - Else, if P_i received at least $t+1$ messages of the form $\langle S_b\rangle_j$ and did not not receive $\langle S_{1-b}\rangle_j$ in round $d+2$, it sets $y_i = b$ and grade $g_i = 1$.
  - Else, if P_i received less than $t+1$, but at least one message of the form $\langle S_1\rangle_j$, it sets $y_i = 1$ and $g_i = 0$.
  - Else, it sets $y_i = 0$ and $g_i = 0$.
  - P_i outputs y_i, g_i, $F^\star_i$ and terminates.

Fig. 3. Code of $\Pi^k_{d\text{-}\mathsf{GDB}}$ for party P_i.

Protocol $\Pi^k_{d\text{-GDA}}$

- **Input and Initialization:** Let v_i, and F_i denote P_i's input. P_i sets $y_i := v_i, g_i := 0, H_i, F_i^\star := \emptyset$, and $y_{i,j}, i_{i,j} := \bot$ for $j \in [n]$.
- **Rounds $r = 1$ to $d+2$:** Party P_i invokes $\Pi^k_{d\text{-GDB}}$ on input (v_i, F_i). For all $j \in [n]$, denote the instance of $\Pi^k_{d\text{-GDB}}$ in which P_j is the sender as $\Pi^{k,j}_{d\text{-GDB}}$ and denote $(y_{i,j}, g_i^j, F_i^j)$ as the output.
- **Output Determination:**
  - Party P_i accumulates the faulty lists of all instances of $\Pi^j_{\text{d-MCoD}}$ along with the input F_i as $F_i^\star = \bigcup_{j\in[n]} F_i^j \cup F_i$.
  - If there exists $v \in \{0,1\}$ and at least $t+1$ instances of $\Pi^{k,j}_{d+1\text{-GDB}}$ that terminate with $g_i^j = 1$ and output $y_{i,j} = v$, then P_i sets the grade $g_i := 1$.
  - P_i sets $y_i := v$, where v is the the majority bit among values $y_{i,j}$ output.
  - Party P_i outputs $y_i, g_i, F_i^\star$ and terminates.

Fig. 4. Code of $\Pi^k_{d\text{-GDA}}$ for party P_i.

Lemma 11. *$\Pi^k_{d\text{-GDA}}$ achieves d-Detecting Graded Agreement as per Definition 4 and terminates in $d+2$ rounds.*

3.3 Byzantine Agreement Protocol Instantiation

Finally, to achieve optimal round complexity, we instantiate the BA protocol in Fig.1 using $\Pi^k_{d\text{-GDA}}$ from Fig.4. We conclude this section by proving the following main theorem.

Theorem 2. *Assume a PKI setup and $t < n/2$. Let $\Pi^k_{d\text{-GDA}}$ be the d-Detecting Graded Agreement protocol in Fig. 4. Then, Π_{BA} achieves Byzantine agreement as per Definition 2. Furthermore, Π_{BA} terminates in $f + 6\lceil\sqrt{f}\rceil + 6$ rounds.*

Proof. Validity and agreement follow from Lemmata 1 and 6. For complexity, note that each iteration runs for $d+2$ rounds and we eliminate d parties per iteration. Thus, $c = 2$. From Lemma 7, Π_{BA} terminates in $f+6\lceil\sqrt{f}\rceil+6$ rounds. □

4 Early Stopping Broadcast in $\min\{f+3, t+1\}$

In this section, we introduce our early-stopping Byzantine broadcast protocol. We begin by discussing key definitions and primitives required for the protocol, and then proceed to describe its construction. Note that the network and setup assumptions, and adversary model remain consistent with those defined in Section 2.

4.1 EIG Preliminaries

For the EIG tree notation, we use a notation similar to the one described in [3].

Exponential Information Gathering Let Σ_r denote set of all possible strings of length r of elements from $[n]$ without repetition, and define $\Sigma = \bigcup_{1 \le j \le t+1} \Sigma_j$. An Exponential Information Gathering tree (see Fig. 5) is a data structure where each node represents an element in Σ, and edges connect a node to the node representing its longest proper prefix. Since this section focuses on the broadcast problem, we set $\Sigma_1 = s$, where P_s is the designated sender and s is the root node of the EIG tree. The root is located at depth 1, and the depth of any other node is defined inductively as one greater than the depth of its parent. A node at depth d has $n - d$ children, meaning the root has $n - 1$ children. The sequence of labels along the path from the root to a specific node uniquely identifies the path. We use the Greek letter σ to denote such a sequence of labels. Each string σ in Σ represents up to $t + 1$ distinct party names and corresponds uniquely to a path in the EIG tree. The last node in the path σ is denoted by $\hat{\sigma}$. To refer to a child of node $\hat{\sigma}$, we use $\widehat{\sigma q}$, where σq is the sequence σ concatenated with $q \in [n]$. We use *depth* of a node $\hat{\sigma}$ to refer to the number of edges on the path from the root node to $\hat{\sigma}$. Similarly, we use *height* to denote the number of edges on the longest path from $\hat{\sigma}$ to a leaf node. The length of a path σ, denoted by $|\sigma|$, matches the depth of node $\hat{\sigma}$. Additionally, the notation $\hat{\sigma_1} < \hat{\sigma_2}$ indicates that $\hat{\sigma_1}$ and $\hat{\sigma_2}$ lie on the same path in the EIG tree, with $|\sigma_1| < |\sigma_2|$. For two sequences σ and σ', we write $\sigma' \subset \sigma$ if σ' is a proper prefix of σ, and $\sigma' \subseteq \sigma$ if σ' is a prefix of σ (possibly equal to σ). Finally, each node $\hat{\sigma}$ in the EIG tree is associated with two variables during the protocol's execution: *(1)* $val(\hat{\sigma}, i) = v$, representing the initial value stored at node $\hat{\sigma}$ in Tree_i, and (2) $val^\star(\hat{\sigma}, i) = v$, representing the final value stored at node $\hat{\sigma}$ at the protocol's conclusion. It is important to note that the final value $val^\star(\hat{\sigma}, i)$ may be equal to the initial value $val(\hat{\sigma}, i)$.

Next, we define the compound signature chain, a key component used in the broadcast protocol. Unlike the signature chain in Definition 6, which consists solely of nested signatures, the compound signature chain allows for a combination of signed messages and nested signatures. Then, we describe the protocol construction and the early resolve rules.

Definition 7 (Compound Signature Chain). *Let $m \in \{0,1\}^*$ and $l \in \mathbb{N}$. Let $j_1, \ldots, j_l$ be distinct values in $[n]$, and let σ_l be ordered sequence of these l values, i.e., $\sigma_l = j_1, \ldots j_l$. A compound signature chain on message m, $\mathsf{cSig}(m, \sigma_l)$, is recursively defined as follows:*

– $\mathsf{cSig}(m, j_1)$*: A tuple of a message m and a signature by party P_{j_1} on m, represented in the form $\langle m \rangle_{\sigma_1}$.*
– $\mathsf{cSig}(m, \sigma_l)$*: A tuple taking one of the two following forms:*
  1. *The tuple $\langle \mathsf{cSig}(m, \sigma_{l-1}), \langle \mathsf{resolve}_m, \mathsf{prefix}_{l-1} \rangle_{j_l} \rangle$, where $j_l \notin \sigma_{l-1}$ and prefix_{l-1} is some prefix of σ_{l-1}.*

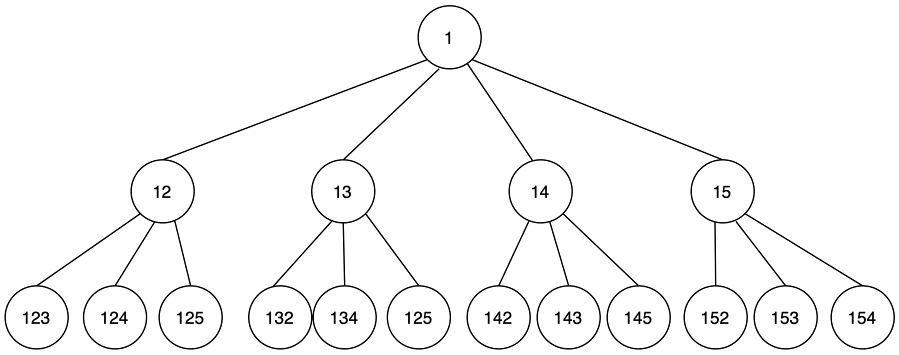

Fig. 5. EIG graph representation for $n = 5$ parties, and $t = 2$ malicious parties. We set party P_1 to be the designated sender. The root node is at depth $d = 1$, and each node at depth d has $n - d$ children. The graph represents the EIG tree by the end of round $t + 1$.

2. $\langle \mathsf{cSig}(m, \sigma_{l-1})\rangle_{j_l}$, *where* $j_l \notin \sigma_{l-1}$.

We recursively define a compound signature chain as *valid* if either it is of the form $\mathsf{cSig}(m, j_1) = \langle m\rangle_{j_1}$ and the signature is valid with respect to public key pk_{j_1} or if it is a syntactically correct (in the above sense) tuple of the form $\mathsf{cSig}(m, \sigma_l)$, where $\mathsf{cSig}(m, \sigma_{l-1})$ is valid, and signature $j_l \in \sigma_l$ is valid with respect to the public key pk_{j_l}.

Notation For simplicity, we henceforth represent a compound signature chain $\mathsf{cSig}(m, \sigma_l)$ as $\langle m\rangle_{\sigma_l}$. here, we override the notation that we previously defined in section 6.

4.2 Protocol Construction

The protocol follows a two-stage paradigm similar to the EIG solution in [26]. In the first stage, parties exchange and forward messages for up to $t+1$ rounds, storing received information in an EIG tree. In the second stage, each party computes its output by determining the *resolved* value of each node through a bottom-up approach. This process begins at the leaf nodes and progresses toward the root, where the resolved value of a node is computed using a predefined function applied to the resolved values of its children. The resolved values of the leaves correspond to those stored during the first stage. Ultimately, each party outputs the resolved value at the root as its final decision.

Our protocol addresses the Byzantine broadcast problem, where there is a designated sender, P_s that sends his value to all parties and parties need to agree on the sender's value. Consequently, node $\hat{s}$ serves as the root of the tree for all parties, and each party ultimately outputs the value to which node $\hat{s}$ is resolved. In other words, each party's final decision corresponds to the resolved value stored in node $\hat{s}$, which they interpret as the agreed-upon value sent by the designated sender. Our protocol is 1-biased, meaning that in the first round, the designated sender P_s sends its initial value to all parties only if $v_s = 1$; otherwise, P_s does not send any message. At the end of the round, each party P_i updates its EIG tree based on the messages it has received. Specifically, if P_i receives $\langle 1 \rangle_s$ from the sender, it sets $val(\hat{s}, i) = \langle 1 \rangle_s$, indicating that node $\hat{s}$ stores the value 1. Otherwise, it sets $val(\hat{s}, i) = 0$. In the subsequent round, parties echo the sender's value by sending $\langle \widehat{si}, \langle val(\hat{s}, i) \rangle_i \rangle$ to all other parties, ensuring that each party P_j records the message received from P_i at node $\widehat{si}$. This process continues in each round following the EIG paradigm, where parties echo and store received values in newly added leaves of the tree or assign a default value of 0 if no message is received.

For the EIG-based protocol to achieve early stopping, parties must cease exchanging information within a number of rounds proportional to the exact number of malicious parties, where $f \leq t$. This is based on the observation that a party P_i can often determine in advance the resolved value of a given node $\hat{\sigma}$ in the EIG protocol. We define this resolved value as $val^\star(\hat{\sigma}, i)$, a process known as *prediction*. Once P_i predicts $val^\star(\hat{\sigma}, i)$, it can stop collecting and forwarding information for the entire subtree rooted at $\hat{\sigma}$. When a party does this, we say that P_i has *closed* node $\hat{\sigma}$. Since many nodes can be resolved before the protocol completes its communication phase, we introduce early resolve rules that allow nodes to be resolved in rounds earlier than $t + 1$.

Next, we start by explaining the output function and how the output value is calculated. Then, we explain the intuition behind our early resolve rules.

4.3 Output Computation Function

A party terminates under one of the following conditions: *(1)* it resolves the root node of the tree by setting $val^\star(\hat{s}, i)$ before round $t + 1$, *(2)* it resolves all the leaves of the tree before round $t + 1$, or *(3)* it reaches the end of round $t + 1$. In case of *(1)*, the party directly outputs the resolved value of the root node. Otherwise, if *(2)* or *(3)* holds, the party executes the Π_{output} subroutine, which is a deterministic function to recursively resolve the nodes from the leaves up to the root. When $t < n/2$, an honest node does not necessarily have a majority of honest children. Consequently, unlike settings where $t < n/3$ or $t < n/4$, the deterministic function cannot rely on a simple majority rule. As our protocol is 1-biased, our deterministic function is also 1-biased. Specifically, if at least $t + 1 - |\sigma|$ of a node $\hat{\sigma}$'s children hold the value 1, party P_i sets $val^\star(\hat{\sigma}) = 1$ at the end of the protocol. As shown in Fig.7, every honest node has at least $t + 1 - d$ honest children, where d is the node's depth. This ensures that if an honest sender initially sends the value 1, then every honest node along any fully honest path will also resolve to 1. Ultimately, all parties will reach agreement on the root node's value being 1. Fig. 6 illustrates the Π_{output} subroutine, which each party executes to determine its final output value, y_i.

```
                         $\Pi_{\mathsf{Output}}$

– **Input and Initialization:** let tree $\mathsf{Tree}_i$ denote $P_i$'s input.
– **Execution:**
    • Each $P_i$ recursively applies the following rules from the leaves to
      the root of the tree $\mathsf{Tree}_i$ for each node $\hat{w}$ where $\mathsf{val}^\star(\hat{w}, i) = \emptyset$:
        ∗ for each node leaf $\hat{w}$, set $\mathsf{val}^\star(\hat{w}, i) := \mathsf{val}(\hat{w}, i)$
        ∗ for each non-leaf node $\hat{w}$, set $\mathsf{val}^\star(\hat{w}, i) = 1$ if $\exists \geq (t+1-(|w|))$
          children with $val^\star(\widehat{wj}, i) = 1$. Else, set $\mathsf{val}^\star(\hat{w}, i) = 0$
– **Output Determination:** $P_i$ returns value $\mathsf{val}^\star(\hat{s}, i)$.
```

Fig. 6. Code of Π_{Output} subroutine.

4.4 Early Resolve Rules

To achieve early stopping in the $t < n/2$ setting, we introduce two key early resolve rules. The first applies to the parents of leaf nodes (nodes of height $h = 1$), while the second extends to nodes at greater heights in the tree (nodes of height $h > 1$). These rules allow a party to close a node in an earlier round once it can accurately predict its final resolved value.

4.4.1 Nodes of height $h = 1$

Resolve rule for value 1. To determine when a node $\hat{\sigma}$ should be resolved to the value 1, denoted as $val^\star(\hat{\sigma}, i) = 1$, we introduce a simple yet powerful rule. For a parent node of leaf nodes (i.e., a node with height $h = 1$), at least $n - 2|\sigma| + 1$ of its children must already hold the value 1 for party P_i to confidently assign $\hat{\sigma}$ the resolved value 1.
Why is this rule safe? Consider the two possible cases for $\hat{\sigma}$: it is either honest or malicious. If $\hat{\sigma}$ is honest, the correctness proof is trivial; because an honest party either sends message 1 to all parties or sends nothing at all. More importantly, a malicious party cannot forge a message claiming $\hat{\sigma}$ sent 1 due to the unforgeability of signatures. We later formally prove that for any honest node $\hat{\sigma}$, all honest parties will resolve it to the same value they received from it, ensuring $val^\star(\hat{\sigma}, i) = val(\hat{\sigma}, i)$.

If $\hat{\sigma}$ is malicious, things get trickier. A malicious $\hat{\sigma}$ could send 1 to some honest parties while withholding it from others. In this case, we must consider two possibilities: either every node along the path σ is malicious, or at least one honest node exists along the path. If an honest node is present, we are safe—honest parties will all resolve it to the same value as mentioned earlier, ensuring agreement. But what if every node in σ is malicious? Even then, the adversary can only corrupt at most $t - |\sigma|$ of its children. For $\hat{\sigma}$ to be resolved early to 1, party P_i must receive at least $n - 2|\sigma| + 1 = 2t + 2 - 2|\sigma|$ children with value 1, ensuring that at least $t + 1 - |\sigma|$ of them are honest. Since all honest parties eventually resolve honest children $\widehat{\sigma j}$ to 1, they will also resolve $\hat{\sigma}$ to 1 through the Π_{output} subroutine.

Why is the rule early stopping? To see why this rule achieves early stopping, consider a path of length f consisting entirely of malicious nodes. Let $\hat{\sigma}$ be the first honest node on this path, where $|\sigma| = f + 1$. Since $\hat{\sigma}$ has at least $n - |\sigma|$ honest children, and $n - |\sigma| \geq n - 2|\sigma| + 1$, enough honest children will echo its value 1. As a result, $\hat{\sigma}$ is resolved in round $|\sigma| + 1 = f + 2$, and closed in round $f + 3$ after echoing on its children.

Now, suppose an honest node $\hat{\sigma}$ exists on path σ, but not all other nodes along σ are malicious. In the worst case, assume that all nodes in σ are honest. This scenario is worst because honest node $\hat{\sigma}$ can have—up to $f \leq t$ malicious children. These malicious children can abstain from sending 1 for node $\hat{\sigma}$, creating a delay in resolving node $\hat{\sigma}$. Here, $\hat{\sigma}$ has the minimum possible number of honest children, given by $n - f - |\sigma|$. If P_i receives fewer than $n - 2|\sigma| + 1$ messages with value 1 in round $|\sigma| + 1$, it implies that at least $f > |\sigma| - 1$ malicious parties abstained from sending value 1. However, this delay does not violate early stopping, as the protocol allows additional rounds to be run when $f > |\sigma|$.

The deepest honest node $\widehat{\sigma\alpha}$ in such a path is resolved when the condition $n - 2|\sigma\alpha| + 1 \geq n - f - (|\sigma\alpha| - 1)$ holds. Simplifying, we get $f = |\sigma\alpha| - 1$, meaning $\widehat{\sigma\alpha}$ is resolved in round $|\sigma\alpha| + 1 = f + 2$. This ensures that the round complexity for resolving any honest node is at most $f + 2$, guaranteeing early stopping as long as $f < t$. Once all the leaves in the tree are resolved, a party can terminate and compute its output, as discussed in section 4.3.

Resolve rule for value 0**.** A party P_i safely resolves node $\hat{\sigma}$ to 0 if it does not receive any message from child $\widehat{\sigma j}$ with value 1. This follows from the Π_{output} function, which requires at least $t + 1 - |\sigma|$ children with value 1 for $\hat{\sigma}$ to be resolved to 1 by the end of the protocol. If all nodes along path σ are malicious and attempt to mislead honest parties by setting $\hat{\sigma}$ to 1 for some and 0 for others, they must rely on at least one honest child to send $val(\hat{\sigma}, i) = 1$ to all parties. However, at most $t - |\sigma|$ nodes can be malicious when all nodes in σ are corrupted. If some nodes in σ are honest, $\hat{\sigma}$ may have more than $t - |\sigma|$ malicious children. For this deception to work, there must be an honest ancestor of $\hat{\sigma}$ with value 1, as otherwise, the malicious parties cannot produce a valid signature chain containing value 1. In this case, all honest parties will resolve this ancestor to 1, ensuring agreement.

Conversely, if $\hat{\sigma}$ is honest and has value 0 (does not send any message), an adversary would need to forge its signature to generate a valid message of the form $\langle 1 \rangle_\sigma$ to mislead parties into resolving $\hat{\sigma}$ differently. Since signature forgery is infeasible, any honest node $\hat{\sigma}$ with value 0 is resolved no later than round $|\sigma|+1$.

4.4.2 Nodes of height $h > 1$

For nodes with height $h > 1$, we only have one rule for the value 1. For value 0, no additional rules are necessary because these nodes can be set within a round as explained in the previous section. To set an node $\hat{\sigma}$ of height $h > 1$ to 1, at least $t + 1 - |\sigma|$ children must be resolved to 1, i.e., $val^\star(\widehat{\sigma j}, i) = 1$. This is

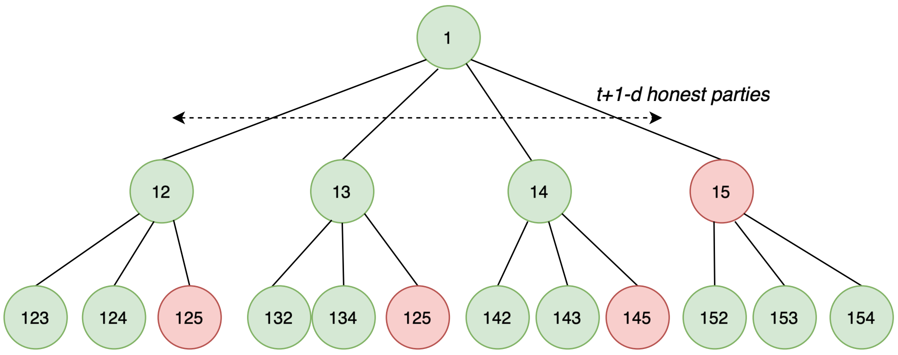

Fig. 7. EIG graph representation for $n = 5$ parties, assuming Party P_1 as the designated sender, with $f = 1$ malicious node. Malicious nodes are represented by ●, while honest nodes are represented by ●. Each honest node has at least $t + 1 - d$ honest children, where d is the node's depth. Given $f = 1$, the earliest honest node (e.g., node 12) to be set using the early resolve rule appears at depth $d = 2$. To set a node with height $h = 2$ to value 1, at least $n - 2|d| + 1$ children must send value 1. As, $n - 2|2| + 1 = 2$, the node is set at the end of round 3 after receiving input from its children. Thus, Party P_5 delayed the setting of honest nodes in the tree by only one round.

the same rule used in Π_{output} at the end of the protocol. In every round of our protocol, the party checks if it can set internal nodes early, recursively checking from the bottom nodes (parents of leaves) up to the root.

Finally, the resolving rules are summarized in Π_{resolve} subroutine, which is invoked by parties within Π_{BB}.

Π_{resolve}

- **Input and Initialization:** let tree Tree_i and path σ denote P_i's input.
- **Execution:**
  - P_i sets $\mathsf{val}^\star(\hat{\sigma}, i)$ according to the following criteria:
    - If $\hat{\sigma}$ is of height $h = 1$:
      - $\mathsf{val}^\star(\hat{\sigma}, i) := 1$ if there exists at least $n - 2(|\sigma|) + 1$ children $\widehat{\sigma j}$ of $\hat{\sigma}$ with $val(\widehat{\sigma j}, i) = \langle 1 \rangle_{\sigma j}$
      - $\mathsf{val}^\star(\hat{\sigma}, i) := 0$ if there is no node $\hat{\sigma j}$ with with $\mathsf{val}(\widehat{\sigma j}, i) = \langle 1 \rangle_{\sigma j}$
    - If $\hat{\sigma}$ is of height $h > 1$:
      - $\mathsf{val}^\star(\hat{\sigma}, i) := 1$ if there exists at least $t + 1 - (|\sigma|)$ children $\widehat{\sigma j}$ of $\hat{\sigma}$ with $\mathsf{val}^\star(\widehat{\sigma j}, i) = 1$
  - **Output Determination:** P_i returns $\mathsf{val}^\star(\hat{\sigma}, i)$

Fig. 8. Code of Π_{resolve} subroutine.

4.5 Future Certificates

If an honest party P_i resolves a node $\hat{\sigma}$ at round r, setting $val^{\star}(\hat{\sigma}, i) \neq \emptyset$ and subsequently closing the subtree rooted at $\hat{\sigma}$, it must ensure that other honest parties will also assign the same value to $\hat{\sigma}$ in a later round $r' > r$. When the resolved value is 0, this process is straightforward, as honest parties interpret the silence of P_i as an implicit confirmation of having received message 0. However, resolving $\hat{\sigma}$ to 1 introduces additional complexities. To assign $\hat{\sigma}$ the value 1, it must satisfy the condition that at least $t+1-|\sigma|$ of its children, denoted as $\widehat{\sigma\alpha}$, have the value 1. Furthermore, these children must each have at least $t+1-|\sigma\alpha|$ of their own children also assigned the value 1, and this requirement continues recursively throughout the subtree.

A key challenge is ensuring that once P_i closes the node $\hat{\sigma}$ and its descendants, the absence of votes from P_i does not prevent other honest parties from resolving $\hat{\sigma}$ to 1. To mitigate this issue, when P_i resolves $\hat{\sigma}$ to 1 at round r, it reports on the leaves of the subtree rooted at $\hat{\sigma}$ in round $r+1$ and broadcasts $\langle \mathsf{resolve}_1, \sigma \rangle_i$ to all honest parties. If an honest party P_j receives $\langle \mathsf{resolve}_1, \sigma \rangle_i$ from P_i at round $r > |\sigma|+1$, it sets $val(\widehat{\sigma k i}, j) = 1$ provided that $val(\widehat{\sigma k}, j) = 1$. Furthermore, P_j then forwards $\langle \mathsf{resolve}_1, \sigma \rangle_i$ to all other parties in the next round.

This certificate serves as a valid future signature from P_i, allowing P_j to construct a legitimate compound chain for the value 1 in a later round, even if P_i has stopped reporting on this subtree rooted at $\hat{\sigma}$. The chain incorporates P_i's signature on the descendants of $\hat{\sigma}$. For example, the resulting chain could be structured as $\langle \langle 1 \rangle_{\sigma k}, \langle \mathsf{resolve}_1, \sigma \rangle_i \rangle_j$, which is a valid chain on 1 for node $\widehat{\sigma k i j}$.

4.6 Early Subtree Closure

After party P_i resolves node $\hat{\sigma}$ in round r, i.e., $val^{\star}(\hat{\sigma}), i) = v$, P_i no longer needs to acquire additional information about the descendants of node $\hat{\sigma}$. This is because it has already predicted the final value stored at node $\hat{\sigma}$, and any additional information about the descendants would not change anything. As a result, P_i reports on the descendants of node $\hat{\sigma}$ of length r for the last time in round $r+1$. It assigns the final value of all descendants $\widehat{\sigma j}$ of $\hat{\sigma}$ to v, such that $val^{\star}(\widehat{\sigma j}, i) = v$. Finally, it stops reporting or storing any subsequent nodes it might receive within this subtree rooted at node $\hat{\sigma}$. At this point, we say that P_i has *closed* node $\hat{\sigma}$ (or equivalently, the subtree rooted at node $\hat{\sigma}$).

```
$\mathit{\Pi}_{\mathsf{close}}$

– Input and Initialization: let tree $\mathsf{Tree}_i$ and path $\sigma$ denote $P_i$'s input.
– Execution:
    • If $\mathsf{val}^\star(\hat{\sigma}, i) = v$, $P_i$ sets $\mathsf{val}^\star(\widehat{\sigma j}, i) := v$ for all descendant nodes $\widehat{\sigma j}$
      in $\mathsf{Tree}_i$, where $|\sigma j| \leq t + 1$
– Output Determination: $P_i$ returns $\mathsf{Tree}_i$
```

Fig. 9. Code of $\mathit{\Pi}_{\mathsf{close}}$ subroutine.

```
$\mathit{\Pi}_{\mathsf{BB}}$

– Input and Initialization: Let $v_s$ denote $P_s$'s input. Each $P_i$ sets
  $y_i := 0, S_i, C_i := \emptyset$. $P_i$ constructs initial EIG $\mathsf{Tree}_i$ with only root node
  $s$. $P_i$ sets $val(\hat{s}, i) = \emptyset$.
– Round 1:
    • If $P_i = P_s$ and $v_s = 1$, $P_i$ sends $\langle 1 \rangle_s$ to all parties. Else, does
      nothing.
    • At the end of the round, if $P_i$ receives $\langle 1 \rangle_s$ from $P_s$, it sets
      $val(\hat{s}, i) := \langle 1 \rangle_s$. Else, it sets $val(\hat{s}, i) := 0$.
– Round $r = 2$ to $r = t + 1$ ($P_i \neq P_s$):
    • $P_i$ sends all messages in $S_i$, and then sets $S_i := \emptyset$.
    • For each path $w$, such that $|w| = r - 1$ and $val^\star(\hat{w}, i)$ was set in
      round $r - 1$, $P_i$ sends $\langle \widehat{wi}, \langle val(\hat{w}, i) \rangle_i \rangle$ to all parties.
    • For each path $w$, such that $|w| = r - 1$ and $val^\star(\hat{w}, i) = \emptyset$, $P_i$ sends
      $\langle \widehat{wi}, \langle val(\hat{w}, i) \rangle_i \rangle$ to all parties.
    • At the end of the round, If $P_i$ receives any message of the form
      $\langle \mathsf{resolve}_1, \sigma \rangle_j$, it adds it to $C_i$, and $S_i$ if never been added before.
    • For each path $w$, such that $|w = \alpha j| = r$ and $val^\star(\hat{w}, i) = \emptyset$:
        ∗ If $val(\hat{\alpha}, i) = \langle 1 \rangle_\alpha$ and $\exists$ $\langle \mathsf{resolve}_1, \sigma \rangle_j$ in $C_i$, such that $\sigma \subset \alpha$,
          $P_i$ sets $val(\widehat{\alpha j}, i) = (\langle 1 \rangle_\alpha, \langle \mathsf{resolve}_1, \sigma \rangle_j)$.
        ∗ Else if, $P_i$ receives $\langle \hat{w}, \langle 1 \rangle_w \rangle$, it sets $val(\hat{w}, i) := \langle 1 \rangle_w$.
        ∗ Else, $P_i$ sets $val(\hat{w}, i) := 0$.
    • Close nodes: $P_i$ checks if it can close some subtrees early by
      invoking subroutine $\mathit{\Pi}_{\mathsf{resolve}}(\hat{\sigma}, \mathsf{Tree}_i)$, followed by $\mathit{\Pi}_{\mathsf{close}}(\hat{\sigma}, \mathsf{Tree}_i)$ on
      each node $\hat{\sigma}$ of the tree $\mathsf{Tree}_i$ recursively in a bottom-up fashion
      starting from leaves.
    • Add future certificates: For all nodes $\hat{\sigma}$ in tree $\mathsf{Tree}_i$, if $\mathsf{val}^\star(\hat{\sigma}, i)$
      is set to 1 in round $r$, then $P_i$ adds $\langle \mathsf{resolve}_1, \sigma \rangle_i$ to $S_i$, if
      $\nexists \langle \mathsf{resolve}_1, \alpha \rangle_i \in S_i$ such that $\alpha \subset \sigma$.
– Output Determination: Terminate if either of the following holds:
  (1) $r = t + 1$, (2) node $s$ is resolved (i.e., $\mathsf{val}^\star(\hat{s}, i) \neq \emptyset$), or (3) all
  leaf nodes are resolved. For the output value, set $y_i = val^\star(\hat{s}, i)$ if it is
  not empty. Else, calculate $y_i$ by running $\mathit{\Pi}_{\mathsf{output}}(\mathsf{Tree}_i)$, and output the
  resultant value.
```

Fig. 10. Code of $\mathit{\Pi}_{\mathsf{BB}}$ for party p_i.

References

1. L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, p. 382–401, jul 1982. [Online]. Available:

https://doi.org/10.1145/357172.357176

2. D. Dolev and C. Lenzen, "Early-deciding consensus is expensive," in *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*, ser. PODC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 270–279. [Online]. Available: https://doi.org/10.1145/2484239.2484269
3. I. Abraham and D. Dolev, "Byzantine agreement with optimal early stopping, optimal resilience and polynomial complexity," in *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, ser. STOC '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 605–614. [Online]. Available: https://doi.org/10.1145/2746539.2746581
4. K. Perry and S. Toueg, "An authenticated byzantine generals algorithm with early stopping," *Cornell eCommons*, 1984, online.
5. F. Elsheimy, J. Loss, and C. Papamanthou, "Early stopping byzantine agreement in $(1+\epsilon)\cdot f$ rounds," Asiacrypt 2024, 2024. [Online]. Available: https://eprint.iacr.org/2024/822
6. M. Fitzi and J. Nielsen, "On the number of synchronous rounds sufficient for authenticated byzantine agreement," in *Distributed Computing.* Berlin, Heidelberg: Springer Berlin Heidelberg, 09 2009, pp. 449–463.
7. P. Berman and J. A. Garay, "Cloture votes: n/4-resilient distributed consensus in t+ 1 rounds," *Mathematical systems theory*, vol. 26, pp. 3–19, 1993.
8. D. Dolev and H. Strong, "Authenticated algorithms for byzantine agreement," *SIAM J. Comput.*, vol. 12, pp. 656–666, 11 1983.
9. D. Dolev, R. Reischuk, and H. R. Strong, "Early stopping in byzantine agreement," *J. ACM*, vol. 37, no. 4, p. 720–741, oct 1990. [Online]. Available: https://doi.org/10.1145/96559.96565
10. P. Berman, J. A. Garay, and K. J. Perry, "Bit optimal distributed consensus," *Computer Science*, pp. 313–321, 1992.
11. J. A. Garay and Y. Moses, "Fully polynomial byzantine agreement in t + 1 rounds," in *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, ser. STOC '93. New York, NY, USA: Association for Computing Machinery, 1993, p. 31–41. [Online]. Available: https://doi.org/10.1145/167088.167101
12. J. Garay and Y. Moses, "Fully polynomial byzantine agreement for n > 3t processors in t + 1 rounds," *SIAM Journal on Computing*, vol. 27, no. 1, pp. 247–290, 1998. [Online]. Available: https://doi.org/10.1137/S0097539794265232
13. P. R. Parvédy and M. Raynal, "Optimal early stopping uniform consensus in synchronous systems with process omission failures," in *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, ser. SPAA '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 302–310. [Online]. Available: https://doi.org/10.1145/1007912.1007963
14. K. Alpturer, J. Y. Halpern, and R. van der Meyden, "Optimal eventual byzantine agreement protocols with omission failures," in *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing*, ser. PODC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 244–252. [Online]. Available: https://doi.org/10.1145/3583668.3594573
15. A. Castañeda, Y. Moses, M. Raynal, and M. Roy, "Early decision and stopping in synchronous consensus: a predicate-based guided tour," in *International Conference on Networked Systems.* Springer, 2017, pp. 206–221.
16. J. Loss and J. B. Nielsen, "Early stopping for any number of corruptions," in *Advances in Cryptology – EUROCRYPT 2024: 43rd Annual International*

Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26–30, 2024, Proceedings, Part III. Berlin, Heidelberg: Springer-Verlag, 2024, p. 457–488. [Online]. Available: https://doi.org/10.1007/978-3-031-58734-4_16

17. G. Deligios, I. Klasovita, and C.-D. Liu-Zhang, "Optimal early termination for dishonest majority broadcast," Cryptology ePrint Archive, Paper 2024/1656, 2024. [Online]. Available: https://eprint.iacr.org/2024/1656
18. P. Feldman and S. Micali, "Optimal algorithms for byzantine agreement," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, ser. STOC '88. New York, NY, USA: Association for Computing Machinery, 1988, p. 148–161. [Online]. Available: https://doi.org/10.1145/62212.62225
19. I. Abraham, T.-H. H. Chan, D. Dolev, K. Nayak, R. Pass, L. Ren, and E. Shi, "Communication complexity of byzantine agreement, revisited," 2020.
20. J. Wan, H. Xiao, S. Devadas, and E. Shi, "Round-efficient byzantine broadcast under strongly adaptive and majority corruptions," in *Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part I.* Berlin, Heidelberg: Springer-Verlag, 2020, p. 412–456. [Online]. Available: https://doi.org/10.1007/978-3-030-64375-1_15
21. J. Katz and C.-Y. Koo, "On expected constant-round protocols for byzantine agreement," in *Advances in Cryptology - CRYPTO 2006*, C. Dwork, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 445–462.
22. J. Wan, H. Xiao, E. Shi, and S. Devadas, "Expected constant round byzantine broadcast under dishonest majority," in *Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part I.* Berlin, Heidelberg: Springer-Verlag, 2020, p. 381–411. [Online]. Available: https://doi.org/10.1007/978-3-030-64375-1_14
23. A. Karlin and A. C. Yao, "Probabilistic lower bounds for byzantine agreement and clock synchronization," in *Unpublished manuscript*, 1984.
24. M. Fitzi, C. Liu-Zhang, and J. Loss, "A new way to achieve round-efficient byzantine agreement," in *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, ser. PODC'21. New York, NY, USA: Association for Computing Machinery, 2021, p. 355–362. [Online]. Available: https://doi.org/10.1145/3465084.3467907
25. D. Ghinea, V. Goyal, and C.-D. Liu-Zhang, "Round-optimal byzantine agreement," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques.* Springer, 2022, pp. 96–119.
26. P. Berman, J. A. Garay, and K. J. Perry, "Optimal early stopping in distributed consensus," in *Distributed Algorithms*, A. Segall and S. Zaks, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 221–237.

A Proof of Correctness of $\Pi^k_{d\text{-}\mathsf{GDB}}$

In this section, we provide full proof of correctness for protocol $\Pi^k_{d\text{-}\mathsf{GDB}}$(Fig 3)

Lemma 12. *If party P_s is in the faulty list of all honest parties at the beginning of the run, all honest parties output $y_i = 0$ and $g_i = 1$*

Proof. From the protocol construction, honest parties accept a message with value 1 only if the message contains a valid signature chain that begins with

P_s's signature and includes valid PoP_s^k. By assumption, the sender P_s is in the faulty list of all honest parties. From Lemma 9, no party can output a valid PoP_s^k from Π_{PoP}^k. Consequently, no party can form a valid chain that includes PoP_s^k. Hence, no honest party P_i ever accepts a message with a value of 1. In round $d+1$, every honest party sends the message $\langle \mathsf{Vote}_0, \mathsf{Nomsg}, \mathsf{PoP}_i^k \rangle$. Malicious parties cannot send $\langle \mathsf{Vote}_1, \langle 1 \rangle_\sigma, \mathsf{PoP}_i^k \rangle$ because $\langle 1 \rangle_\sigma$ must be a valid signature chain. By the end of round $d+2$, each honest party receives at least $t+1$ messages of the form $\langle \mathsf{Vote}_0, \mathsf{Nomsg}, \mathsf{PoP}_i^k \rangle_i$. In round $d+2$, each honest party compiles a set S_0 containing $t+1$ of these received messages and broadcasts S_0 to all other parties. Due to the unforgeability of signatures, a malicious party cannot construct a valid set S_1. Thus, by the end of round $d+2$, every honest party receives at least $t+1$ messages containing the set S_0. Consequently, each honest party outputs $y_i = 0$ with a grade of $g_i = 1$.

Lemma 13. *If all honest parties receive $\langle 1 \rangle_\sigma$ by the end of round d, all honest parties output $y_i = 1$ and $g_i = 1$. On the other hand, if no honest party receives $\langle 1 \rangle_\sigma$ by the end of round $d+1$, all honest parties output $y_i = 0$ and $g_i = 1$.*

Proof. We will prove each separately. Let all honest parties receive $\langle 1 \rangle_\sigma$ by the end of round d. In round $d+1$, each honest party P_i sends $\langle \mathsf{vote}_1, \langle 1 \rangle_\sigma, \mathsf{PoP}_i^k \rangle_i$ to all other honest parties. As a result, every honest party receives at least $t+1$ messages of the form $\langle \mathsf{vote}_1, \langle 1 \rangle_\sigma, \mathsf{PoP}_j^k \rangle_j$ and constructs the set S_1. In round $d+2$, each honest party sends S_1 to all other parties. Since there are at most $t < n/2$ malicious parties, no honest party receives the set S_0. Therefore, every honest party receives at least $t+1$ sets of S_1 and outputs $y_i = 1$ with $g_i = 1$. Now, assume that no honest party receives $\langle 1 \rangle_\sigma$ by the end of round $d+1$. Thus, in the beginning of round $d+1$, each honest party P_i sends $\langle \mathsf{vote}_0, \mathsf{Nomsg}, \mathsf{PoP}_i^k \rangle_i$ to all other parties. By the end of round $d+1$, each honest party receives at least $t+1$ messages of the form $\langle \mathsf{vote}_0, \mathsf{Nomsg}, \mathsf{PoP}_i^k \rangle_i$. Moreover, no honest party receives $\langle \mathsf{vote}_1, \langle 1 \rangle_\sigma, \mathsf{PoP}_i^k \rangle_j$ by assumption. Consequently, in round $d+2$, each honest party sends the set S_0 to all other parties. No malicious party can send a valid set S_1, due to signature unforgeability. Thus, every honest party receives at least $t+1$ messages of the form S_0 and outputs $y_i = 0$ with $g_i = 1$.

Lemma 14. *If no honest party receives $\langle 1 \rangle_\sigma$ by the end of round d, then every honest party P_j outputs $y_j = 0$.*

Proof. By assumption, every honest party sends a message of the form $\langle \mathsf{vote}_0, \mathsf{Nomsg}, \mathsf{PoP}_i^k \rangle_i$, and no honest party sends $\langle \mathsf{vote}_1, \langle 1 \rangle_\sigma, \mathsf{PoP}_i^k \rangle_i$ during round $d+1$. For an honest party P_j to output $y_j = 1$ in round $d+2$, it must receive at least one message of the form $\langle S_1 \rangle_j$, where S_1 is a set containing at least $t+1$ messages of the type $\langle \mathsf{vote}_1, \langle 1 \rangle_\sigma, \mathsf{PoP}_j^k \rangle_j$. Since the number of malicious parties is at most t , the adversary cannot generate such a set S_1 without forging signatures. Therefore, every honest party sets $y_j = 0$.

Lemma 15. *If an honest party receives a chain, $\langle 1 \rangle_{l_1, \cdots, l_r}$, for the first time in the protocol in round r, then $P_{l_1}, \ldots, P_{l_{r-1}}$ must be malicious.*

Proof. According to the protocol, if an honest party P_j receives a chain for the first time in round r, it will forward it to all parties in round $r+1$. Therefore, if an honest party receives a chain $\langle 1\rangle_{l_1,\cdots,l_r}$ for the first time in round r, it knows that parties $P_{l_1},\ldots,P_{l_{r-1}}$ must be malicious; otherwise, it would have received the chain in an earlier round.

Lemma 16. *If honest parties P_i and P_j output $y_i \neq y_j$, then there exists a non-empty proper subset of honest parties, H_1, that receives a chain on message 1 in round d.*

Proof. By Lemma 13, honest parties output the same value if either *(a)* all honest parties receive $\langle 1\rangle_\sigma$ by the end of round d, or *(b)* no honest party receives $\langle 1\rangle_\sigma$ by the end of round $d+1$. Thus, honest parties may output conflicting values only if *(a)* only a proper subset of honest parties, H_1, receive $\langle 1\rangle_\sigma$ by the end of round d, or *(b)* a non-empty subset of honest parties receive a chain, $\langle 1\rangle_\sigma$, for the first time in the protocol by the end of round $d+1$. However, in the latter case, by Lemma 14, every honest party P_j outputs $y_j = 0$. Therefore, conflicting outputs among honest parties can only occur if a non-empty proper subset of honest parties, H_1, receive $\langle 1\rangle_\sigma$ by the end of round d. Note that H_1 must be non-empty; otherwise, it would be identical to scenario *(b)*.

Lemma 17. *Let honest parties P_i and P_j output $y_i \neq y_j$. Then, at least d common malicious parties are added to $F_i^\star$ and $F_j^\star$ by the end of round $d+1$.*

Proof. From Lemma 16, honest parties P_i and P_j can output $y_i \neq y_j$ only if a non-empty proper subset of honest parties, H_1, receive a chain on message 1 in the end of round d. We show that all honest parties will detect at least d malicious parties. For parties in H_1, each honest party adds the first $d-1$ malicious parties, $P_{l_1},\ldots,P_{l_{d-1}}$, to their faulty lists by Lemma 15. They then append their own signature to the chain $\langle 1\rangle_{l_1,\cdots,l_d,i}$ and forward it to all parties in round $d+1$. If any party in H_1 receives at least $t+1$ messages of the form $\langle \mathsf{vote}_1, \langle 1\rangle_\sigma, \mathsf{PoP}_i^k\rangle_i$ in round $d+1$, it constructs set S_1 and forwards these messages to all parties in round $d+2$, ensuring that all honest parties receive them and set $y_i = 1$. Otherwise, if a party in H_1 receives fewer than $t+1$ such messages, it adds P_{l_d} to its faulty list, as P_{l_d} must have failed to send $\langle 1\rangle_{l_1,\cdots,l_d}$ to all honest parties in round d, confirming its malicious behavior. For parties outside H_1, they receive $\langle 1\rangle_{l_1,\cdots,l_d,i}$ for the first time in round $d+1$ and add all d malicious parties, $P_{l_1},\ldots,P_{l_d}$, to their faulty lists. Thus, every honest party detects at least d malicious parties if conflicting outputs occur.

We now give the proof of Lemma 10.

Proof (Of Lemma 10). Assume that for each honest party P_i, $P_i \notin F_j$ for any honest party P_j.
Graded Validity: Suppose that the sender P_s is honest. In the first round, if P_s's initial value is 1, it sends this value to all parties; otherwise, it sends nothing. By assumption, P_i is not in the faulty list of any honest party. According to Lemma 8, every honest party P_i outputs PoP_s^k from Π_{PoP}^k. Let $v_s = 1$.

Then, every honest party receives $\langle 1 \rangle_s$ in the first round. Every honest party constructs $\langle \langle 1 \rangle_s, \mathsf{PoP}_s^k, \mathsf{PoP}_i^k \rangle_i$ and forwards it to all parties in the next round. Consequently, from Lemma 13, all honest parties output $y_i = 1$ with $g_i = 1$. Otherwise, if $v_s = 0$, then P_s sends nothing to any party in round 1. In round $d+1$, each honest party P_i sends $\langle \mathsf{vote}_0, \mathsf{Nomsg}, \mathsf{PoP}_i^k \rangle_i$ to all other parties. By the end of the round, each honest party has received at least $t+1$ messages of the form $\langle \mathsf{vote}_0, \mathsf{Nomsg}, \mathsf{PoP}_j^k \rangle_j$. Moreover, no honest party receives $\langle \mathsf{vote}_1, \langle 1 \rangle_\sigma, \mathsf{PoP}_j^k \rangle_j$, where $\langle 1 \rangle_\sigma$ is a valid chain due to signature unforgeability. Consequently, in round $d+2$, each honest party sends the set S_0 to all other parties. Thus, every honest party receives at least $t+1$ messages of the form S_0 and outputs $y_i = 0$ with $g_i = 1$.

Graded Consistency: Consider two honest parties, P_i and P_j. Assume that P_i outputs $y_i = v$ and $g_i = 1$. We will show that P_j also outputs $y_i = v$. First, suppose $v = 1$. This implies that P_i received messages of the form $\langle S_1 \rangle_k$ from at least $t+1$ parties in round $d+2$, with at least one of these messages coming from an honest party who also sent this message to all other honest parties. Consequently, every honest party must have received at least one message of the form $\langle S_1 \rangle_k$ in round $d+2$, leading them to output $y_i = 1$. Now, consider the case where $v = 0$. In this scenario, P_i must have received at least $t+1$ messages of the form $\langle S_0 \rangle_k$, including at least one message from an honest party P_k in round $d+2$. Additionally, P_i did not receive any $\langle S_1 \rangle_l$ messages. This implies that P_k did not received any $\langle \mathsf{vote}_1, \langle 1 \rangle_\sigma \rangle_l$ messages in round $d+1$, as otherwise, it would not have sent $\langle S_0 \rangle_k$ in round $d+2$. Thus, no honest party could have sent $\langle \mathsf{vote}_1, \langle 1 \rangle_\sigma \rangle_l$ in round $d+1$, as otherwise P_k would have it received by the end of that round, leading to a contradiction. Given that there are at most $t < n/2$ malicious parties, the adversary cannot construct a set S_1 with at least $t+1$ messages of the form $\langle \mathsf{vote}_1, \langle 1 \rangle_\sigma \rangle_l$ to send in round $d+2$. Therefore, no honest party P_j receives a message of the form $\langle S_1 \rangle_k$ in round $d+2$, and all honest parties output $y_i = 0$.

d-Detection: From Lemma 17, if two honest parties, P_i and P_j, output conflicting values ($y_i = 1$ and $y_j = 0$, respectively), then at least d malicious parties are added to the faulty list, $F_i^\star$, of each honest party P_i by the end of round $d+1$.

Soundness: By assumption, at the beginning of the protocol, each honest party P_i is not in the faulty list, $P_i \notin F_j$, for any other honest party P_j. An honest party P_i adds another party P_j to its faulty list under the following conditions:

1. Let P_i receive a message on 1 for the first time in round r. If the message is of the form $\langle 1 \rangle_{l_1, \cdots, l_d k}$, then it is guaranteed that parties $P_{l_1}, \cdots P_{l_d}$ are malicious, as honest parties are required to forward messages on 1 to all other parties as soon as they receive it.
2. P_i receives a message on 1 directly from P_j in the form $\langle 1 \rangle_{\sigma j}$ during round d. However, fewer than $t+1$ parties subsequently send $\langle \mathsf{vote}_1, \langle 1 \rangle_{\sigma'}, \mathsf{PoP}_i^k \rangle_i$ in round $d+1$. This scenario guarantees that P_j is faulty, as a genuinely honest P_j would have broadcast $\langle 1 \rangle_{\sigma j}$ to all honest parties during round d. As a result, honest parties only add corrupted parties to their faulty lists.

Termination: The protocol runs for $d+2$ synchronous rounds

We finally prove Lemma 11.

Proof (Of Lemma 11). Assume that for each honest party P_i, $P_i \notin F_j$ for any honest party P_j.
Graded Validity: Assume all honest parties have the same input value $v_i = v$. In the first round, each honest party P_i invokes as the sender, $\Pi^k_{d\text{-GDB}}$ on input (v, F_i). According to the graded validity of $\Pi^k_{d\text{-GDB}}$(Definition 3), if P_i is honest, each honest party P_j outputs $g^i_j = 1$, and $y_{j,i} = v$. Thus, since there are at most $t < n/2$ malicious parties, every honest party P_i will output (v, g_i, F^j_i) from at least $t+1$ instances of $\Pi^k_{d\text{-GDB}}$. Consequently, each honest party sets $y_i = v$ as v is the majority value and $g_i = 1$.
Graded Consistency: A party P_i sets $g_i = 1$ if at least $t+1$ instances of $\Pi^{k,j}_{d\text{-GDB}}$ terminate with $g^j_i = 1$, and these instances have the same output value $y_{i,j} = v$. Since $t < n/2$, P_i sets $y_i = v$. Let P_i be an honest party that sets $g_i = 1$ and $y_i = v$. From the consistency property of $\Pi^k_{d\text{-GDB}}$, every other honest party P_l outputs $y_{l,j} = v$ for the same instances where P_i outputs $g^j_i = 1$. Thus, every party P_l sets $y_l = y_i$ as the output is determined using majority function. As a result, all honest parties set $y_l = v$.
d-Detection: For two honest parties, P_i and P_j, outputting conflicting values ($y_i = 1$ and $y_j = 0$, respectively), there exists at least one instance j among the n parallel instances of $\Pi^k_{d\text{-GDB}}$ where P_i and P_j output conflicting values. If the sender of $\Pi^k_{d\text{-GDB}}$ is honest, then from graded validity of $\Pi^k_{d\text{-GDB}}$, every honest party outputs the same value. If the sender is a party in the faulty list of all honest parties, then every honest party outputs 0 from Lemma 12. Therefore, the d-Detection property of $\Pi^k_{d\text{-GDB}}$ implies that at least d malicious parties are added to every honest party P_i's faulty list $F^\star_i$ via F^j_i.
Soundness: The output faulty list $F^\star_i$ is constructed as the union of all faulty lists F^j_i, for $j \in l$, generated by the l parallel executions of $\Pi^k_{d\text{-GDB}}$. By the soundness property of the $\Pi^k_{d\text{-GDB}}$ protocol, it follows that the resulting faulty list $F^\star_i$ includes only malicious parties.
Termination: $\Pi^k_{d\text{-GDA}}$ consists of concurrent instances of $\Pi^k_{d\text{-GDB}}$. Based on the assumption that $\Pi^k_{d\text{-GDB}}$ terminates, $\Pi^k_{d\text{-GDA}}$ will also terminate.
Round Complexity: $\Pi^k_{d\text{-GDA}}$ consists of concurrent execution of $\Pi^k_{d\text{-GDB}}$, which runs in $d+2$ rounds. □

B Proof of Correctness of Π_{BB}

Lemma 18. *If P_i and P_j are both honest, then for all nodes $\hat{w}$, $val(\widehat{wj}, i) = val(\hat{w}, j)$.*

Proof. An honest party sends the same message to all parties.

Lemma 19. *Let P_i and P_j be two honest parties. If $val(\hat{w}, i) = 0$, each honest party P_j sets $val^\star(\widehat{wi}, j) = 0$ by at most round $|wi|+1$ and close the subtree with root $\widehat{wi}$ in round $|wi|+1$.*

Proof. As P_i is honest, it sends the same message to all honest parties. Due to the unforgeability of signatures, no malicious party P_k can send $\langle 1 \rangle_{ik}$. According to the Π_{resolve} rule for setting $val^\star(\widehat{wi}, j) = 0$, all honest parties P_j set $val^\star(\widehat{wi}, j) = 0$ once they receive the children of node $\widehat{wi}$ in round $|wi|+1$ and close the subtree with root $\widehat{wi}$ in round $|wi| + 2$.

Lemma 20. *Assume no node is resolved on path σ by the end of round $|\sigma|$ for any honest parties, i.e., $val^\star(\alpha, i) = \emptyset\ \forall \alpha \in \sigma$. Let P_i and P_j be two honest parties. If P_i sets $val^\star(\hat{\sigma}, i) = 0$, then either P_j sets $val^\star(\hat{\sigma}, i) = 0$ by the end of the protocol, or (2) there $\exists$ at least one honest node $\hat{\beta} \neq \hat{\sigma}$ on path σ with value 1 in every honest party's tree.*

Proof. If node $\hat{\sigma}$ is honest, the proof follows directly from Lemma 19. Now, let's consider the case where node $\hat{\sigma}$ is malicious. For an honest party P_i to resolve $\hat{\sigma}$ to 0 in round $|\sigma|+1$, it must not receive any child of $\hat{\sigma}$ with a value of 1. For party P_j to resolve $\hat{\sigma}$ to 1, it must either (1) receive at least $n - 2|\sigma| + 1 > t + 1 - |\sigma|$ children with a value of 1 or (2) resolve at least $t+1-|\sigma|$ children to 1 according to the early resolve rules. If every node in σ is malicious, then there are at most $t - |\sigma|$ malicious children of $\hat{\sigma}$. Therefore, for P_j to set $val^\star(\hat{\sigma}, j) = 1$, it must receive $val(\widehat{\sigma k}, j) = 1$ from at least one honest party for either resolve rules, which leads to a contradiction since party P_i cannot then set $val^\star(\hat{\sigma}, i)$ to 0. Thus, for the adversary to carry out this attack, node $\hat{\sigma}$ must have more than $t - |\sigma|$ malicious children. Hence, there must be at least one honest node $\hat{\beta}$ in path σ, which must have a value of 1; otherwise, the adversary cannot forge its signature and send different values to parties. Since party $\hat{\beta}$ is honest, it sends 1 to all honest parties. From the assumption that no node is resolved (thus not closed) on path σ by the end of round $|\sigma|$ for any honest parties and since $\hat{\beta} < \hat{\sigma}$, $val(\hat{\beta}, j) = 1$ for every honest party P_j

Lemma 21. *Assume no node on path w is resolved for any honest party by the end of round $|w|$, i.e., $val^\star(\alpha, i) = \emptyset\ \forall \alpha \in w$. Let P_i and P_j be two honest parties. For any node $\hat{w}$ such that $val(\hat{w}, j) = 1$, all honest parties P_i sets $val^\star(\widehat{wj}, i) = 1$ by the end of the protocol.*

Proof. We need to demonstrate that honest parties always observe enough descendants to set $val^\star(\widehat{wj}, i) = 1$. We distinguish two scenarios (1) P_i sets $val^\star(\widehat{wj}, i) = 1$ during the protocol execution; or (2) all honest parties P_i set $val^\star(\widehat{wj}, i) = 1$ at the end of the protocol using the Π_{output} subroutine. We will start by proving the second scenario.

According to Lemma 18, party P_j sends the same message, $\langle 1 \rangle_{wj}$, to all honest parties. We then prove the lemma by induction on $t+1-|wj|$. If $|wj| = t+1$, then $val^\star(\widehat{wj}, i) = val(\hat{w}, j)$ as per the leaf node setting rule in Π_{output}. If $|wj| < t+1$, then following the non-leaf node rule in Π_{output}, a party P_i sets $val^\star(\widehat{wj}, i)$ to (1) if there exist at least $t+1-|wj|$ children with value $val^\star(\widehat{wjk}, i) = 1$, or to (2) 0 otherwise. By the inductive hypothesis, $val^\star(\widehat{wjk}, i) = val(\widehat{wj}, k)$, which equals

$val(\hat{w}, j)$ from Lemma 18. Note that node $\widehat{wj}$ has at least $n - |wj|$ children, of which at least $t+1-|wj|$ are possibly honest since node $\widehat{wj}$ is not fixed by any honest node, and thus the chain is not closed earlier. These honest nodes have $val^\star(\widehat{wjk}, i) = val(\hat{w}, j) = \langle 1 \rangle_j$.

Now, we prove the 1st scenario. Assume that some honest party P_k resolves node $\widehat{wj}$ during the protocol execution. This could happen if it uses the Π_{resolve} function in round $r < t+1$. Consequently, in the next round party P_k sends $\langle \mathsf{resolve}_i, \widehat{wj} \rangle_k$ to all honest parties. Thus, all honest parties set $val(\widehat{wj\sigma k}, i) = 1$ for any node σ with $val(\sigma, i) = 1$, where σ is a descendant of $\widehat{wj}$. Therefore, for every honest node $\widehat{wj\sigma}$, it has at least $t+1-|wj\sigma|$ children with $val(\widehat{wj\sigma}, i) = 1$ in every honest parties' trees, either from future certificates or from an honest party that has not yet terminated.

Lemma 22. *Assume no node on path σ is resolved for any honest party by the end of round $|\sigma|$, i.e., $val^\star(\alpha, i) = \emptyset \ \forall \alpha \in \sigma$. Let $\hat{\sigma}$ be a node that's a parent of leaves. Let P_i be an honest party that resolves node $\hat{\sigma}$ to 1 such that $val^\star(\hat{\sigma}, i) = 1$ in round $|\sigma|+1 < t+1$, then either (1) every other honest party P_k also resolves $\hat{\sigma}$ to 1 by setting $val^\star(\hat{\sigma}, k) = 1$ by the end of the protocol, or (2) there $\exists$ at least one honest node $\hat{\beta} \neq \hat{\sigma}$ on path σ with value 1 in every honest party's tree.*

Proof. If node $\hat{\sigma}$ is honest, then every honest party P_k sets $val^\star(\hat{\sigma}, k) = 1$ based on Lemma 21. Now, consider the scenario where node $\hat{\sigma}$ is malicious. There are two possibilities: either every node in σ is malicious, or there exists an honest node $\hat{\beta}$ in σ.

First, assume every node in σ is malicious. We will show that all honest parties resolve $val^\star(\hat{\sigma}, k) = 1$. A party P_i sets $val^\star(\hat{\sigma}, i) = 1$ if it receives values of 1 from at least $n - 2|\sigma| + 1$ children of $\hat{\sigma}$. Since $n - 2|\sigma| + 1 = 2t + 2 - 2|\sigma|$, there must be at least $t+1-|\sigma|$ honest children, because at most $t-|\sigma|$ children can be malicious. According to Lemma 21, all honest parties resolve at least $t+1-|\sigma|$ honest children of $\hat{\sigma}$ to a value of 1 by the end of the protocol, such that $val^\star(\widehat{\sigma j}, k) = 1$. Consequently, $val^\star(\hat{\sigma}, k) = 1$ is set either through the Π_{output} subroutine or the early resolve rules.

If not every node in $\hat{\sigma}$ is malicious, there must be an honest node $\hat{\beta}$ in σ. This honest node $\hat{\beta}$ must have a value of 1 or else malicious node $\hat{\sigma}$ can not send a valid message on 1 to party P_i without party $\hat{\beta}$'s signature. From the assumption that no node is resolved (and thus, closed) on path σ by the end of round $|\sigma|$ for all honest parties and since $\hat{\beta} < \hat{\sigma}$, $val(\hat{\beta}, j) = 1$ for every honest party P_j.

Corollary 2. *(Validity) If P_s is honest and starts with value $v_s = v$, all honest parties will output v.*

Proof. If party P_s has $v_i = 1$, it sends $\langle 1 \rangle_s$ to all parties. From lemma 21 assuming $w = \emptyset$, it follows that all honest parties will set $val^\star(s, i) = 1$ and $y_i = 1$ by the end of the protocol. Otherwise, if party P_s has $v_i = 0$, it does not send anything to parties in the first round. From Lemma 19, all honest parties set $val^\star(s, i) = 0$ by the end of round 2.

Lemma 23. *Let P_i be the first honest party that sets $val^\star(\hat{\sigma}, i) = v$ in round r, then no node on the path σ is resolved by any honest party by the end of round $r-1$.*

Proof. By contradiction. Assume honest party $P_l \neq P_i$ set $val^\star(\hat{\beta}, i) = v$, where $\hat{\beta}$ is a node on path σ. P_l would also set the descendants of $\hat{\beta}$ to the same value in round $r-1$, and $\hat{\sigma}$ is a descendant of $\hat{\beta}$. So, honest party P_l would have set $val^\star(\hat{\sigma}, l)$ by the end of round $r-1$, which contradicts the assumption.

Lemma 24. *Let P_i be the first honest party that resolves node $\hat{\sigma}$ to 1 such that $val^\star(\hat{\sigma}, i) = 1$ by the end of round $r < t+1$, then either (1) every other honest party P_k also resolves $\hat{\sigma}$ to 1 by setting $val^\star(\hat{\sigma}, k) = 1$ by the end of the protocol, or (2) there $\exists$ at least one honest node $\hat{\beta} \neq \hat{\sigma}$ on path σ with value 1 in every honest party's tree.*

Proof. From the protocol construction, at round r, the maximum depth a node can have is r, which is a leaf. Let $\hat{\sigma}$ be a node that is resolved at round $r < t+1$. Thus, $\hat{\sigma}$ resolved by one of the early resolve rules as the last round in the protocol is $t+1$. There are two early resolve rules: (1) a rule for nodes with height $h = 1$ (2) a rule for nodes with height $h > 1$. Thus, only nodes $\hat{\sigma}$ at depth $d < r$ can be resolved by the early resolve rules in round $r < t+1$. We prove the lemma by induction on subtrees with root node $\hat{\sigma}$ at depth $d < r$.

Base case $|\sigma| = d = r-1$: If $\hat{\sigma}$'s depth is $r-1$, then it has height $h = 1$ in round r. Furthermore, since P_i is the first party to resolve $\hat{\sigma}$ in round $|\sigma|+1$, no node on path σ is resolved by the end of round $|\sigma|$ for any honest party. And thus, no honest party closed the tree with subroot $\hat{\sigma}$ in round $|\sigma|+1$. Let $\hat{\sigma}$ be resolved to 1 by party P_i in round r such that $val^\star(\hat{\sigma}, i) = 1$. From Lemma 22, every other honest party P_k either also resolves $\hat{\sigma}$ to 1 by setting $val^\star(\hat{\sigma}, k) = 1$ by the end of the protocol, or (2) there $\exists$ at least one honest node $\hat{\beta} \neq \hat{\sigma}$ on path σ with value 1 in every honest party's tree.

Base case $|\sigma| = d = r-2$: If $\hat{\sigma}$'s depth is $r-2$, then it has height $h > 1$ in round r. According to the early resolve rule for such node, $\hat{\sigma}$ is resolved to 1 if at least $t+1-|\sigma|$ of $\hat{\sigma}$'s children are resolved to 1, implying that $val^\star(\hat{\sigma}j) = 1$ in round r, where j refers to some child of $\hat{\sigma}$. These $t+1-|\sigma|$ children have a length of $|\sigma|+1 = r-1$, and they can only be set by the corresponding early stopping resolve rule for nodes with height $h = 1$. Furthermore, they can only be resolved in round $|\sigma j|+1 = r$. From the assumption, we know that party P_i is the first honest party that resolves node $\hat{\sigma}$ in round $|\sigma|+2$. From Lemma 23, no node on the path σ is resolved by any honest party by the end of round $|\sigma|+1$. This further implies that no node on the path σj is resolved by the end of round $|\sigma j| = |\sigma|+1$ since $|\sigma j|$ is a leaf, and there is no early resolve rule for leaves. Consequently, from the base case $d = r-1$, if a party P_i resolves a parent of leaf nodes $\widehat{\sigma j}$ to 1 in round $|\sigma j|+1$, then either (1) every other honest party P_k sets $val^\star(\widehat{\sigma j}, k) = 1$, or (2) there exists at least one honest node $\hat{\beta}$ on the path σj with a value of 1. If the former holds for all $t+1-|\sigma|$ children of $\hat{\sigma}$, then every party P_k sets $val^\star(\hat{\sigma}, k) = 1$ because $t+1-|\sigma|$ children are resolved to 1. If the latter, then either $\hat{\sigma} = \hat{\beta}$ or $\hat{\beta} < \hat{\sigma}$. Assume $\hat{\sigma} = \hat{\beta} = \delta l$, where P_l is an honest node and $val(\delta, l) = 1$. Additionally, no node on the path β was resolved by any honest party by the end of round $|\beta|$, based on the assumption that P_i is the first party to set node $\hat{\sigma}$ in round $r > |\sigma| > |\beta|$. Therefore, $val^{\star}(\hat{\sigma}, i) = val^{\star}(\hat{\sigma}, k) = 1$ follows from Lemma 21, where $\delta = w$. If $\hat{\beta} \neq \hat{\sigma}$, then there exists at least one honest node $\hat{\beta}$ on the path σ with a value of 1 in every honest party's tree as $\hat{\sigma}$ is honest and sends the same value to all parties. This completes the proof for the base cases.

Inductive hypothesis: assume for $d \leq r - 2$, the lemma holds for all subtrees with root nodes $\hat{\sigma}$ with depth $|\sigma| \geq d$. We will show that the lemma holds for subtrees with root nodes $\hat{\sigma}$ with depth $|\sigma| = d - 1$.

Inductive step: $\hat{\sigma}$ is an internal node with depth $|\sigma| = d - 1 < r - 2$ and thus, is not a parent of leaf node. Thus, party P_i sets node $\hat{\sigma}$ to 1 in round r if there are $t + 1 - |\sigma|$ children, $\widehat{\sigma j}$, that are resolved to 1 in round r according to resolve rule for nodes with height $h > 1$. Let's call these $t + 1 - |\sigma|$ children set T_1. Assume these nodes $\widehat{\sigma j}$ in T_1 are first resolved by an honest party in rounds $s_1 \leq \ldots s_{|T_1|} \leq r < t + 1$ respectively. Then, by inductive hypothesis, either other honest parties P_k set $val^{\star}(\widehat{\sigma j}, k) = 1$ by the end of the protocol, or there exists an honest node $\hat{\beta}$ on path σj with value 1 in every honest party's tree. If the former case holds for all nodes, $\widehat{\sigma j}$, in T_1, meaning honest party P_k resolves $val^{\star}(\widehat{\sigma j}, k) = 1$, then honest P_k sets $val^{\star}(\hat{\sigma}) = 1$. If the former does not hold for all nodes in T_1, then there exists at least one honest node $\hat{\beta}$ on path σj with value 1 by the inductive hypothesis. Similar to the base case, either $\hat{\sigma} = \hat{\beta}$ or $\hat{\beta} < \hat{\sigma}$. If $\hat{\sigma} = \hat{\beta}$, then $val^{\star}(\hat{\sigma}, i) = val^{\star}(\hat{\sigma}, k) = 1$ follows from Lemma 21. If $\hat{\beta} < \hat{\sigma}$, then there exists at least one honest node $\hat{\beta} \neq \hat{\sigma}$ on the path σ with value 1.

Corollary 3. *Let $\hat{\sigma}$ be the earliest node (i.e., the node with the smallest depth) on a path that's set by an honest party P_i, such that $val^{\star}(\hat{\sigma}, i) = v$ in round $r < t + 1$. Then, one of the following two conditions holds:* (1) *either all honest parties P_k set it to the same value, $val^{\star}(\hat{\sigma}, k) = v$ or* (2) *there exists an honest node $\hat{\beta} < \hat{\sigma}$ on path σ in every honest party's tree with value 1.*

Proof. Let r be the round in which an honest party P_i sets the node $\hat{\sigma}$ during the protocol execution in $r < t + 1$. Since there is no early resolve rule for a leaf node, r must be greater than $|\sigma|$ ($r > |\sigma|$). Based on the assumption that $\hat{\sigma}$ is the earliest node on path $(\sigma\alpha)$, where $|\alpha| \geq 0$, no node on the path of σ is resolved by any honest party by the end of round $|\sigma|$.

Consequently, P_i sets $val^{\star}(\hat{\sigma}, i) = v \in \{0, 1\}$ in round $|\sigma| + 1 \leq r < t + 1$ by applying one of the early resolve rules. If $v = 0$, then the only early resolve rule for value 0 can be applied in round $|\sigma| + 1$, and according to Lemma 20, two scenarios arise: (1) all honest parties P_k set $val^{\star}(\hat{\sigma}, k) = 0$ by the end of the protocol, or (2) there exists at least one honest node $\hat{\beta} < \hat{\sigma}$ on path σ with value 1 in every honest party's tree. Similarly, if $v = 1$, then according to Lemma 24, the same scenarios arise: (3) all honest parties P_k set $val^{\star}(\hat{\sigma}, k) = 1$ by the end of the protocol, or (4) there exists at least one honest node $\hat{\beta} < \hat{\sigma}$ with value 1 in every honest party's tree.

Lemma 25. *(Agreement) If an honest party P_i outputs $y_i = v$, every honest party P_j outputs $y_i = v$*

Proof. A party P_i sets its output based on the value it resolves the root node $\hat{s}$ to, i.e., outputs v if $val^{\star}(\hat{s}, i) = v$. The root node is either (1) resolved by a subset of honest parties during the protocol run in round $r < t+1$ by using early resolve rules, or (2) all honest parties resolves $\hat{s}$ at the end of the protocol by executing Π_{output} in the end of round $t+1$. We distinguish both cases.

1. Let P_i be the first party that sets $val^{\star}(\hat{s}, i)$ during the protocol run using early resolve rules. If $val^{\star}(\hat{s}, i) = 1$, then based on Lemma 24, every honest party P_j either sets $val^{\star}(\hat{s}, j) = 1$ or there exists an honest node $\hat{\beta} \subset \hat{s}$ with value 1. However, the latter case can not happen as $\hat{s}$ is the first node of the tree. Thus, every honest party P_j sets $val^{\star}(\hat{s}, j) = 1$. On the other hand, let P_i set $val^{\star}(\hat{s}, i) = 0$. The only available early resolve rule for 0 is applied for nodes with height $h = 1$ and thus, P_i resolves $\hat{s}$ at round $|\hat{s}| + 1 = 2$ by setting $val^{\star}(\hat{s}, i) = 0$. From Lemma 20, P_j also sets $val^{\star}(\hat{s}, j) = 0$ for the same reason that it does not exist a node $\hat{\beta}$ before node $\hat{s}$.

2. Assume all honest parties determine $val^{\star}(\hat{s}, i)$ by running Π_{output} at the end of round $t+1$. The function Π_{output} is applied recursively from leaves to root, resolving nodes that were not resolved during the protocol execution. Specifically, Π_{output} resolves nodes as follows:
   - If a node $\hat{\sigma}$ is a leaf, it is set to the value it stored during the protocol execution, i.e., $val^{\star}(\hat{\sigma}, i) = val(\hat{\sigma}, i)$.
   - For a non-leaf node $\hat{\sigma}$ that was not resolved during the protocol execution, an honest party runs a deterministic function over the resolved children of $\hat{\sigma}$. Node $\hat{\sigma}$ is resolved to 1 if at least $t + 1 - |\sigma|$ children are resolved to 1, and 0 otherwise.
   - If node $\hat{\sigma}$ was resolved in some round $r < t+1$, do nothing

   Consequently, we prove agreement by following these two steps:
   - Step 1: We slice the trees, T_i (tree formed during protocol execution before running Π_{output}), of all honest parties P_i horizontally at different lengths, effectively cutting some chains to a shorter length but ensuring all honest parties' trees are cut at the same length (same node) for each chain. After such slicing, we consider the resultant trees T_i' for honest parties P_i. We show that the leaves of these trees have the same resolved value, i.e., $val^{\star}(\hat{\sigma}, i) = val^{\star}(\hat{\sigma}, j)$ for the resultant trees T_i' and T_j' of any two honest parties P_i and P_j. In other words, all honest parties P_i resolves every node $\hat{\sigma}$, where $\hat{\sigma}$ is a leaf in trees T_i', to the same value in trees T_i during the execution of Π_{output}.
   - Step 2: Considering the resultant trees T_i' of honest parties P_i, we demonstrate that after applying Π_{output} on these trees, all honest parties will set node s to the same value, and thus, output the same value.

As you can see, we divide the agreement proof for applying the Π_{output} subroutine into two steps. In the first step, we focus on a selected set of nodes, with one node chosen from each chain. We demonstrate that after running Π_{output} from the leaves up to these nodes, all honest parties resolve these nodes to the same value.In the second step, we consider the trees formed by treating these nodes as leaves. We then show that when Π_{output} is applied to these trees, all honest parties assign the same value to the root node s. Thus, by proving Steps 1 and 2, we prove agreement. We begin with Step 1. For each honest party's tree T_i, where T_i is the tree constructed during the protocol execution, consider the subtree T_i', where T_i' is the same as T_i but with some chains cut at a shorter length. By "cutting a shorter length," we mean the following: let $\sigma_1, \sigma_2, \ldots, \sigma_{t+1}$ be chain c in tree T. If we cut the chain at node σ_3, then the resultant chain in tree T' is $\sigma_1, \sigma_2, \sigma_3$, where σ_3 is now the leaf of chain c in tree T'. We denote the resultant chain as $c[\sigma_3]$. We will show that for each honest party P_i and P_j, for all leaves $\hat{\sigma} \in T_i', T_j'$, it holds that $val^\star(\hat{\sigma}, i) = val^\star(\hat{\sigma}, j)$. We first demonstrate how T_i' is constructed. The tree T_i' is constructed by cutting the chains of tree T_i as follows:
For each chain c in tree T_i for all $i \in [n]$,

- if there exists a node $\hat{\sigma}$ that is the earliest node on chain c resolved by some honest party P_k in some round $r < t + 1$, then
  - If there exists an honest node $\hat{\beta} < \hat{\sigma}$ on path σ with value 1, such that $val(\hat{\beta}, k) = 1$, cut the chain of T_i at $\hat{\beta}$, resulting in the chain $c[\beta]$ in T_i' for all honest parties P_i.
  - If there is more than one honest node with value 1, pick the earliest node.
  - Otherwise, cut the chain at $\hat{\sigma}$, resulting in the chain $c[\hat{\sigma}]$ in T_i' for all honest parties P_i.
- Else, every other chain c in T_i remains the same in T_i'.

Thus, all honest parties' trees are cut at the same length for each chain because all honest parties' EIG trees have the same node locations. Consequently, we show that for all the leaves in the resultant trees T_i', each honest party P_i resolves them to the same value. Note that the leaves for all trees T_i' for $i \in [n]$ fall into the following categories:

(a) An honest node $\hat{\beta}$ with value 1, $val(\hat{\beta}, i) = 1$, where no node δ on path β was resolved by any honest party in any round r, where $|\beta| < r < t + 1$.
(b) A node $\hat{\sigma}$ that is the earliest node set by some honest P_i in some round $r < t + 1$, $val^\star(\hat{\sigma}, i) = v$. Furthermore, there does not exist any honest node $\hat{\beta}$ with value 1, such that $\beta \subset \sigma$.
(c) A node σ where $|\sigma| = t + 1$.

To demonstrate that each leaf in T_i' and T_j' is resolved to the same value for honest parties P_i and P_j, we consider each type separately:

- Type 1: Let $\hat{\beta} = \delta l$, where P_l is an honest node. So, $val(\delta l, i) = val(\delta, l) = 1$ for every honest party P_i as P_l sends the same value to all honest parties. Note that no node on the path β is resolved by any honest party by the end of round $|\beta|$ according to the chain cutting rules. Thus, according to Lemma 21, all honest parties P_i set $val^{\star}(\hat{\beta}, i) = 1$, where $\delta = w$ in Lemma 21's statement.

- Type 2: If a leaf is of this type, then by Corollary 3, $val^{\star}(\hat{\sigma}, i) = val^{\star}(\hat{\sigma}, j)$ for every honest party P_i and P_j as there doesn't exist an honest node $\hat{\beta}$ with value 1 according to the chain cutting rules, where $\beta \subset \sigma$
- Type 3: Honest party P_i sets $val^{\star}(\hat{\sigma}, i) = val(\hat{\sigma}, i)$ according to the Output() function for leaves as stated earlier. Since there are at most t malicious parties, every leaf node of length $t+1$ is honest. Thus, for each leaf node $\widehat{\sigma j}$ of length $t+1$, all honest parties P_i set $val^{\star}(\widehat{\sigma j}, i) = val(\widehat{\sigma j}, i) = val(\hat{\sigma}, j)$ based on Lemma 18.

Hence, we proven Step 1. Now, we prove Step 2. For party P_i to calculate $val^{\star}(s, i)$, each honest party runs the Output() subroutine. This subroutine resolves nodes recursively from the leaves up to the root by applying the deterministic function mentioned earlier. By Step 1, for all leaves in T_i' and T_j' for every honest party P_i and P_j, it holds that $val^{\star}(\hat{\sigma}, i) = val^{\star}(\hat{\sigma}, j)$. Therefore, all honest parties P_i apply the same deterministic function over the same initial input (leaves' resolved values) recursively until they reach the root node in tree T_i'. As a result, the output after applying this recursive deterministic function is the same for all honest parties. Consequently, $val^{\star}(s, i) = val^{\star}(s, j)$ for honest parties P_i and P_j.

Lemma 26. *Let node $\widehat{\sigma j}$ be the first honest node on the path $\sigma j \alpha$, where $|\alpha| \geq 0$. If $|\sigma| \leq t-1$, then the node $\widehat{\sigma j}$ is resolved by all honest parties, such that $val^{\star}(\widehat{\sigma j}, i) = v$ by the latest round $r = \max(|\sigma j| + 1, f - |\sigma| + 2)$. Additionally, all honest parties closes node $\widehat{\sigma j}$ by at most round $r+1$. If $|\sigma| = t$, then the node $\widehat{\sigma j}$ is resolved by round $t+1$.*

Proof. We distinguish between $\sigma \leq t-1$ and $\sigma = t$

- $\sigma \leq t-1$: As P_j is honest, $val(\widehat{\sigma j}, i) = val(\hat{\sigma}, j)$ holds true in all honest parties P_i's trees. If $val(\hat{\sigma}, j) = 0$, then all honest parties set $val^{\star}(\widehat{\sigma j}, i) = 0$ by at most round $|\sigma j|+1$ and close the subtree with root $\widehat{\sigma j}$ by round $|\sigma j|+2$, as per Lemma 19.
  Now, consider the case where $val(\hat{\sigma}, j) = 1$. Let subtree T be the tree with subroot $\widehat{\sigma j}$. To resolve node $\widehat{\sigma j}$ to 1 early, P_i applies Π_{resolve} on node $\widehat{\sigma j}$. If $\widehat{\sigma j}$ is a parent of a leaf node, that's the height of tree T is 2, and the early resolve rule for parent of leaf nodes is applicable, then it is resolved in round $|\sigma j|+1$, and we are done. Otherwise, $\widehat{\sigma j}$ is a non-parent of leaf internal node. The early resolve rule for such nodes states that at least $t+1-|\sigma j|$ children must be resolved to 1, i.e., $val^{\star}(\widehat{\sigma jk}, i) = 1$ to resolve $\widehat{\sigma j}$ to 1. To resolve these children, $\widehat{\sigma jk}$, to 1, at least $t+1-|\sigma jk|$ of their children must be resolved to 1, and so on recursively until reaching the parent of leaf nodes. These parent nodes are resolved using a different resolve rule than other

nodes with height $h > 1$. To set the parent of leaf nodes using the early resolve rule, there must be $n - 2|\sigma j| + 1$ children with value 1.
Now, let's consider subtree T, with height greater than 2. We prove the lemma by showing this subtree can reach at most height $f - |\sigma| + 1$. Note that since node $\widehat{\sigma j}$ is the first honest node on the path by assumption, there are at most $f - |\sigma|$ malicious children on any honest node that's a descendant of node $\widehat{\sigma j}$. Thus, node $\widehat{\sigma j}$ and its honest descendants $\widehat{\sigma j\alpha}$ have at least $n-(f-\sigma)-|\sigma j| > t+1-|j|$ and $n-(f-\sigma)-|\sigma j\alpha| > t+1-|j\alpha|$ honest children, respectively, as f is at most t. By honest descendants $\widehat{\sigma j\alpha}$, we mean that each node in the path $j\alpha$ is honest. Additionally, all the honest descendants $\widehat{\sigma j\alpha}$ of node $\widehat{\sigma j}$, have value $val(\widehat{\sigma j\alpha}) = 1$. Let T_1 be a subtree of T, consisting of all honest chains, where every node in the chain is honest, in subtree T. As shown, every node $\widehat{\sigma j\alpha}$ in T_1 has at least $n-(f-\sigma)-|\sigma j\alpha| > t+1-|j\alpha|$ honest children and value $val(\widehat{\sigma j\alpha}) = 1$. We need to show that the latest round in which all parents of leaf nodes, $\widehat{\sigma j\alpha} \in T_1$, will be set is $r = (f - |\sigma|) + 1$.
Let $\widehat{\sigma j\beta}$ be a parent of leaf node in T_1 in round $|\sigma j\beta|+1$. For an honest party to set $val^\star(\widehat{\sigma j\beta}) = 1$ during the protocol run, at least $n-2|\sigma j\beta|+1$ children of $\widehat{\sigma j\beta}$ must have $val(\widehat{\sigma j\beta k}) = 1$. Thus, at the latest, the first honest node $\hat{\beta}$ that can be set by honest parties in round $|\sigma j\beta| + 1$ occurs when when the following condition holds: $n - (f - |\sigma|) - |\sigma j\beta| = n - 2|\sigma j\beta| + 1$. That means $2|\sigma j\beta| - |j\beta| = f + 1$, so $|\sigma j\beta| = f - |\sigma| + 2$. Consequently, parties report on the children of $\widehat{\sigma j\beta}$ in round $|\sigma j\beta| + 2$.
Thus, every node in tree T_1 will be resolved to 1 in round $|\sigma j\beta| + 1$ by recursively resolving the honest nodes by applying Π_{resolve} for internal nodes. Consequently, $val^\star(\widehat{\sigma j}, i) = 1$ in round $r = \max(|\sigma j|+1, f-|\sigma|+2)$. Parties then recursively call Π_{close} on every node in the tree in a bottom-up manner. According to Π_{close}, for every $val^\star(\widehat{\sigma j}, i) = v$, party P_i sets $val^\star(\widehat{\sigma jk}, i) := v$ for all descendant nodes $\widehat{\sigma jk}$ in $Tree_i$, where $|\sigma jk| \leq t + 1$. As a result, parties stop reporting or storing any subsequent nodes they might receive within this subtree rooted at node $\widehat{\sigma j}$, as every node $\widehat{\sigma jk}$ in this subtree has $val^\star(\widehat{\sigma jk}) \neq \emptyset$. This is because, as shown in Fig 10, parties only send or receive nodes within the subtree rooted at $\hat{\sigma}$ if $val^\star(\hat{\sigma}, i) \neq \emptyset$."

- $|\sigma| = t$: Then, node $\widehat{\sigma j}$ is a leaf of length $t + 1$. Thus, every honest party P_i runs Output() at the end of round $t + 1$ and set $val^\star(\widehat{\sigma j}, i) = val(\widehat{\sigma j}, i)$

Lemma 27. *(Early Stopping) Protocol Π_{BB} terminates in* $min(f + 3, t + 1)$

Proof. The protocol runs for at most $t+1$ rounds. We are left to prove the $f+3$ rounds. From Lemma 26, once there is an honest party j on the chain, the chain with subroot $\widehat{\sigma j}$ closes by at most round $max(|\sigma j| + 2, f - |\sigma| + 3)$. $|\sigma|$ could be at most f, that's that chain of malicious parties. Such chain terminates by at most round $f + 3$.