Optimal Early-Stopping Broadcast via Exponential Information Gathering

Giovanni Deligios[1], Fatima Elsheimy[2], Simon Holmgaard Kamp[3], Ivana Klasovita[4], Chen-Da Liu-Zhang[4], Julian Loss[3], and Charalampos Papamanthou[2]

[1]ChainSecurity
[2]Yale University
[3]Ruhr University Bochum
[4]Lucerne University of Applied Sciences and Arts

Abstract

Early-stopping broadcast protocols guarantee termination based on the actual number of malicious parties, $f \leq t$, encountered during execution, rather than the worst-case scenario of $t < n$ corruptions. The seminal work of Dolev, Reischuk, and Strong [JACM'90] established that $\min\{f + 2, t + 1\}$ rounds are necessary. The influential works of Berman, Garay, and Perry [WDAG'92] and Abraham and Dolev [STOC'15] introduced the first protocols achieving this optimal bound in the $t < n/3$ (unconditional, no setup) regime, while Perry and Toueg [Tech. Report'84] gave the first protocol for the $t < n/2$ setting (with plain PKI), achieving $2 \cdot \min\{f + 2, t + 1\}$ rounds. Despite this early progress, closing the gap to the optimal bound in the computational setting has remained a longstanding open problem, and only recently has progress been made. The best known result in the $t < n/2$ regime is due to Elsheimy, Loss, and Papamanthou [Eprint'25], achieving $f + O(\sqrt{f})$ rounds.[1] In the dishonest-majority regime, the best known result is due to Deligios, Klasovita, and Liu-Zhang [Eurocrypt'25], who obtain the first asymptotically optimal protocol, with $O(f)$ round complexity, for $t = (1 - \epsilon)n$, where $\epsilon > 0$ is a constant.

In this work, we advance the state of the art in two directions. First, in the $t < n/2$ setting with plain PKI, we present a broadcast protocol in which all parties output within optimal $\min\{f + 2, t + 1\}$ rounds. Second, we complete the picture by introducing the first protocol that is optimally resilient ($t < n$) and asymptotically matches the lower bound for any number of corruptions $t < n$. By leveraging exponential information gathering (EIG) graphs, our constructions establish that these round bounds are achievable, providing the first feasibility result for optimal-round protocols, albeit at the cost of exponential communication complexity.

1 Introduction

Byzantine broadcast is a fundamental problem in distributed computing, first introduced in the seminal work of Lamport, Shostak, and Pease in the 1980s [24]. A broadcast protocol [25, 28, 10] enables a designated sender to disseminate a message to all parties such that two key properties are satisfied: *agreement*, meaning all honest parties (i.e., parties that follow the protocol) decide on the same value, and *validity*, meaning that if the sender is honest, the decided value equals the sender's

[1]That work also gives a protocol achieving decision within $\min\{f + 2, t + 1\}$ rounds, followed by an additional communication round before termination. The present submission subsumes and strengthens that result by achieving termination within $\min\{f + 2, t + 1\}$ rounds.

input. These guarantees must hold even in the presence of up to t faulty or malicious parties, that can deviate from the protocol.

Broadcast is used as a central primitive underlying more complex protocols such as multi-party computation (MPC) and verifiable secret sharing (VSS). Consequently, a large body of work has focused on improving its efficiency across different network models and adversarial settings.

1.1 Round Complexity of Broadcast

A primary measure of efficiency is the *round complexity*, which is defined as the maximum number of successive rounds of information exchange required in any protocol execution.

The classical lower bound of Dolev, Reischuk, and Strong [13] showed that $\min\{f+2, t+1\}$ rounds are necessary for any broadcast protocol, where f is the *actual* number of corrupted parties in a given execution. This bound highlights the exciting possibility of *early-stopping* protocols, whose round complexity scales with the actual number of faults f (rather than scaling with the worst-case corruption threshold t [14, 30]).

A major conceptual milestone in the study of round complexity for Byzantine broadcast is the *Exponential Information Gathering (EIG)* paradigm introduced by Pease, Shostak, and Lamport [28]. The EIG approach provides a conceptually simple and robust solution achieving optimal resilience and number of rounds, but at the cost of exponential communication complexity. Despite this inefficiency, it has served as a crucial stepping stone: many later works build on its structure and ideas to obtain more efficient protocols, reducing communication from exponential to polynomial. Primary examples are the protocols of Dolev and Strong [14] and Pfitzmann and Waidner [30], which reduced the communication of [28] to polynomial.

This recipe also carried over to early-stopping protocols. In the unauthenticated setting (unconditional, no setup), Berman, Garay and Perry [5] proposed the first EIG-based protocol that is optimal in both the number of rounds and the corruption threshold $t < n/3$. This served as a basis for the improvement due to Abraham and Dolev [2], who further achieved optimal round complexity while incurring polynomial communication complexity, fully resolving this setting.

The picture in the authenticated setting (where protocols assume a plain PKI for signatures[2] and are possible for $t \geq n/3$ corruptions) is much less understood, and existing work has focused mainly on achieving asymptotically optimal round and polynomial communication complexity.

In the honest-majority $t < n/2$ regime, the best known protocols incur a multiplicative factor with respect to round optimality. Perry and Toueg [29] incur a factor-2 overhead in rounds. This was improved by Elsheimy, Loss, and Papamanthou [17] to a $(1+\epsilon)$ factor, for constant $\epsilon > 0$, and more recently [18] to $f + O(\sqrt{f})$ rounds. Their protocol relies on iteratively invoking a primitive that detects and collectively eliminates d malicious parties in $d+2$ rounds, where the additional two rounds are used for parties to exchange their views from the preceding d rounds and thereby agree on the same set of faulty parties. To obtain $f+O(1)$ rounds using this approach, however, parties would need to *collectively* identify the same d faulty parties within d rounds, despite potentially receiving inconsistent information across rounds. In particular, it remains open whether such coordination is achievable, and more broadly, whether one can attain round optimality up to additive factors, i.e. $\min\{f + O(1), t+1\}$ in this regime.

The gap becomes even more pronounced in the dishonest-majority regime $t \geq n/2$. The work of Loss and Nielsen [26] achieves optimal resilience $t < n$ but requires $O(\min\{f^2, t+1\})$ rounds. This was very recently improved by Deligios, Klasovita and Liu-Zhang [9], who obtain an asymptotically optimal $O(f)$ rounds (up to a multiplicative constant factor) only for suboptimal

[2] In a plain PKI, each party independently generates and registers its own key pair with no trusted party certifying correct key generation; consequently, corrupted parties may choose their keys adversarially (see [6] for discussion).

Setting	Threshold	Termination rounds	Reference
No setup	$t < n/4$	$\min\{f+3, t+1\}$	[4]
	$t < n/3$	$\min\{f+2, t+1\}$	[5, 2]
Plain PKI	$t < n/2$	$2 \cdot \min\{f+2, t+1\}$	[29]
		$(1+\epsilon)f + O(1)$	[17]
		$f + O(\sqrt{f})$	[18, 16]
		$\min\{f+2, t+1\}$	**Our work**
	$t < (1-\epsilon)n$	$O(f+1)$	[9]
	$t < n$	$O(\min\{f^2+1, t+1\})$	[26, 9]
		$9f + 7$	**Our work**

Table 1: Summary of early-stopping broadcast protocols. Here $\epsilon > 0$ is a constant. The bound $9f + 7$ applies for $f \geq 1$. The protocol of [9] tolerates any $t < n$, with the linear round bound applying when $t < (1-\epsilon)n$.

resilience $t = (1-\epsilon)n$. Thus, no existing protocol simultaneously achieves optimal resilience and asymptotically optimal round complexity.

Given the current state of the art and the need for new techniques, we propose to take a step back. Instead of targeting polynomial communication, we ask whether it is possible to achieve round-optimal constructions in the authenticated setting using the EIG paradigm.

Assuming a plain PKI, can we use the EIG-paradigm to obtain round-optimal protocols that are resilient in the $t \geq n/3$ regime?

1.2 Contributions

We advance the state of the art via the EIG paradigm. Similar to prior results in this regime [17, 26, 9], our constructions only rely on a plain PKI setup, require no number theoretic assumptions, and are secure against strongly adaptive adversaries.

We first present an early-stopping broadcast protocol that tolerates $t < n/2$ corruptions and guarantees that every honest party terminates within the optimal $\min\{f+2,\ t+1\}$ rounds, where f is the actual number of corrupted parties.

Theorem 1 (Informal)**.** *Let n and $t < n/2$ be natural numbers. Assuming a plain PKI, there exists a broadcast protocol tolerating up to t corruptions and such that, if f is the actual number of corruptions in an execution, then every honest party terminates within* $\min\{f+2, t+1\}$ *rounds.*

Furthermore, we give the first early-stopping broadcast protocol which tolerates any number $t < n$ of corruptions and is asymptotically round optimal, meaning that our protocol runs in $O(f)$ rounds.

Theorem 2 (Informal)**.** *Let n and $t < n$ be natural numbers. Assuming a plain PKI, there exists a broadcast protocol tolerating up to t corruptions and such that, if $f \geq 1$ is the actual number of corruptions in an execution, then the protocol runs in $9f + 7$ rounds.*

Notably, these optimal round bounds were not previously known to be achievable even with trusted setup providing correlated randomness, whereas our protocols achieve them assuming only a plain PKI.

1.3 Technical Overview

This paper develops two early-stopping broadcast protocols that operate in different resilience regimes and, accordingly, rely on two distinct ways of reasoning about information in the system. At a high level, both protocols are guided by the same principle: a party should terminate as soon as it can argue that no future information can change its decision. The difficulty, of course, is that in the presence of Byzantine behavior, it is not obvious when this point has been reached. The two protocols approach this question from different angles.

The first part of the paper presents an early-stopping broadcast protocol for $t < n/2$ that outputs in $\min\{f + 2, t + 1\}$ rounds and builds directly on the seminal Exponential Information Gathering (EIG) paradigm. The second part of the paper presents a different early-stopping broadcast protocol for the general $t < n$ setting with round complexity $9f + 7$ for $f \geq 1$. In contrast to the first result, this construction does not follow the standard EIG template and instead relies on a new structural approach.

1.3.1 Warm-up: How EIG Achieves Early Stopping when $t < n/3$ [4, 2]

Overview. A common starting point for early-stopping agreement is the EIG framework [25]. At a high level, the EIG approach systematically records how information propagates through the network by organizing it into a tree structure that captures "who said what about what" across rounds. Each party constructs this tree locally as the protocol progresses. The protocol proceeds in rounds, where in each round parties propagate the information they received in the previous round. Concretely, in the first round, the sender broadcasts its value, and parties store this value in the root node. In the second round, each party reports to all others the value it received from the sender, and parties store these values in the children of the root. In subsequent rounds, parties iteratively report what they heard from others. More generally, in round r, each party forwards the values it received in round $r - 1$, appending its own identifier to the corresponding path. In this way, each path from the root to a leaf reflects a unique sequence of parties, corresponding to a chain of message forwarding. This process continues for $t + 1$ rounds.

At the end of round $t+1$, once this information-gathering phase is complete, each party computes an output by *resolving* its local EIG tree in a bottom-up manner. Informally, resolving refers to assigning a final value to every node in the tree, starting from the leaves and proceeding upward, based on the values of its children. Concretely, leaf nodes take the values received in the final round, and every internal node is assigned a value as a function of its children. In the $t < n/3$ setting, this is typically done via recursive majority: each node takes the majority value among its children, breaking ties toward a fixed default. A fundamental drawback of the EIG approach is that the tree frontier grows rapidly with the number of rounds, leading to exponential communication in the worst case.

Making it Early Stopping. To achieve early stopping round complexity, these protocols introduce *early resolving* rules that allow parties to *prune* subtrees whose eventual resolved values are already determined. These rules take advantage of the fact that parties can often predict the resolved values of some nodes well before the protocol completes, and thus, can stop reporting information on the tree in a number of rounds that is proportional to the exact number of malicious parties in the system, $f \leq t$. In the honest-majority setting $t < n/3$, this pruning argument typically relies on the fact that, for every node whose children set is sufficiently large, the children contain an honest majority. However, for the more challenging $t < n/2$ setting, this property no longer holds, complicating the early stopping rules. Finally, to obtain polynomial communication, both [4] and [2]

add a step called *cloture voting*. The idea is that the early-stopping EIG protocol terminates within a constant number of rounds c whenever all honest parties start with the same initial value; if it does not terminate within c rounds, the parties instead terminate quickly with a default value. This approach relies on two important properties of the underlying EIG protocol. First, whenever all honest parties start with the same input, they output that value within two rounds, regardless of the behavior of corrupted parties. Consequently, the auxiliary EIG instances can terminate quickly whenever their honest inputs agree, preventing their communication from growing exponentially. Second, a long execution in which the EIG tree continues to grow must repeatedly expose new faulty parties. These parties become commonly identified by all honest parties within a constant number of rounds and can subsequently be excluded from the EIG computation. Thus, either the protocol terminates quickly, or continued execution makes progress by identifying faulty parties and reducing the number of parties that can contribute to the growth of the tree.

These properties, however, do not directly carry over to our construction for $t < n/2$. In particular, even with an honest sender, Byzantine behavior may prevent termination within two rounds. Moreover, our early-termination condition identifies a small cover of the observed conflicts, rather than a common set of faulty parties. Consequently, adapting cloture voting to this setting would require additional techniques to control the growth of the EIG tree and achieve polynomial communication.

1.3.2 An Early-Stopping Broadcast Protocol when $t < n/2$

When we move to the $t < n/2$ regime, the earlier early-stopping reasoning breaks down: nodes in the EIG tree may no longer have an honest majority among their children. In fact, a node far down a long chain may have only a single honest child among many faulty ones. As a result, even when all honest parties behave consistently, their signal can be overwhelmed by adversarial behavior, and the usual majority-based resolution argument fails. Our construction takes a different approach. Rather than deciding when to close individual subtrees, a party checks whether the inconsistencies it has observed can all be accounted for by a small set of parties. When this condition holds, it computes its output from the messages collected so far and terminates. The main difficulty is to ensure that parties using different amounts of information still compute the same output.

From signed messages to conflicts. The communication follows the EIG approach. In round 1, the sender P_s signs and sends its input bit to every party. In each subsequent round, every active party signs and sends to all parties a list of the exact messages it received in the preceding round, using $\perp$ whenever no valid message was received. Each party stores the received messages in its local EIG tree, whose paths record what parties said they received from one another. Each party P_i also maintains a conflict graph with one vertex per party and initially no edges. These edges record inconsistencies that P_i discovers by comparing the messages it received directly with what others claim to have received. Suppose, for example, that P_i received m from P_j in some round, but P_k's signed list in the next round states that it received something different from P_j, possibly nothing at all. Either P_j behaved inconsistently toward the two recipients, or P_k is not describing what it received truthfully. In either case, P_j and P_k cannot both be honest. Party P_i therefore adds an edge $\{P_j, P_k\}$ to its graph, without having to determine which party is responsible.

Other observations identify a faulty party directly and are recorded as self-loops. For instance, if P_i sees two different messages signed by P_j for the same round, it adds the self-loop $\{P_j, P_j\}$. An honest party would have sent the same signed message to every party, so this establishes that P_j is faulty. Party P_i also checks the sequence of rounds in which it receives messages from each party. Every honest party sends in round 2 and continues sending until it terminates, after which it never

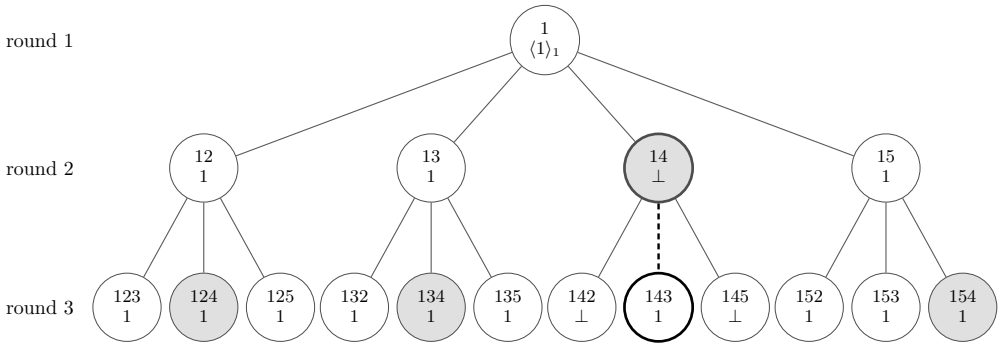

Figure 1: The EIG tree constructed by P_2 for $n = 5$ and $t = 2$, with honest sender P_1 holding 1 and faulty party P_4 (Gray nodes). Each node shows a path and the sender receipt recorded in the corresponding signed report. In round 2, P_4 sends P_2 and P_3 different signed messages. Both become visible to P_2 by round 3, at nodes $\widehat{14}$ and $\widehat{143}$, proving that P_4 equivocated. Thus P_2 adds $\{P_4, P_4\}$ to $\mathsf{Conflicts}_2$.

sends again. Thus, if P_i receives no valid round-2 message from P_j, or receives a message from P_j after a gap, it adds the self-loop $\{P_j, P_j\}$. Similarly, if P_i receives no valid message from the sender in round 1, it adds $\{P_s, P_s\}$.

Stopping when the conflicts have a small cover. At the end of round $2 \le r < t + 1$, a party P_i checks whether there is a set $S \subseteq \mathcal{P} \setminus \{P_i\}$ of at most $r - 2$ parties that intersects every conflict pair. Thus, S is a vertex cover of its conflict graph. If such a cover exists, P_i computes its output and terminates. Otherwise, it continues running the test every round until it reaches round $t + 1$, where it computes the output and terminates.

A cover is not a set of parties that have all been proved faulty. For example, the conflicts $\{P_a, P_b\}$ and $\{P_a, P_c\}$ can both be covered by $\{P_a\}$, even if the actual faulty parties are P_b and P_c. Nor do honest parties need to find the same cover. Each party uses only the conflicts visible in its own tree. This gives a simple argument for early termination. Every conflict observed by an honest party contains a faulty endpoint. Consequently, the set F of all parties corrupted during the execution covers every honest party's conflicts. If $|F| = f$, then this cover meets the size bound by round $f + 2$. Any party that has not terminated earlier computes its output and terminates at the end of round $t + 1$. Together, these two termination rules ensure that every honest party terminates within $\min\{f + 2, t + 1\}$ rounds. It remains to explain why a small cover also allows a party to choose an output safely.

Computing the output. The output depends on the parties' round-2 messages, which state what they received from the sender in round 1. A faulty party, however, may have sent different round-2 messages to different parties. The output function uses the later signed messages to determine which of these messages contribute to the output. The computation proceeds from the leaves of the EIG tree toward the root. At a node of depth d ending at P_j, the output function considers P_j's messages from rounds $2, \ldots, d$, one round at a time. For each round u, it computes a set of signed round-u messages of P_j, denoted $\mathrm{val}^{(u)}(\hat{w}, i)$. At a leaf, this set initially contains only the message that P_i received directly from P_j in round u, or is empty if no valid message was received. At a node ending at P_i itself, P_i uses its own signed message. At every other internal node, P_i uses the entries already computed at the children. Consider a signed round-u message m from P_j that appears in P_i's tree. A child ending at P_k supports m if its round-$(u + 1)$ entry states that it received exactly m from P_j in round u. If at least $n - t - d$ distinct children support m, then P_i

includes m in the node's round-u entry. Otherwise, it does not include that message.

Why do we require $n - t - d$ supporting children? First consider the case where no honest party has yet terminated. A path of length d has at least $n - t - d$ honest children that are not on the path. If P_j is honest, all these parties receive its message and include it in their next message. This ensures that honest messages have enough support at every level of the computation. Now consider a path whose d parties are all faulty. At most $t - d$ faulty parties remain among its children. Since $n > 2t$, we have $n - t - d > t - d$. Thus, the faulty children alone cannot provide enough support for a message to be included. At least one supporting child must be honest, and it sends that message to every honest party. Applying this procedure up to the root's children determines which round-2 messages are used to compute the output bit. Each child is assigned 1 if its round-2 entry contains a signed message giving the sender's signed value 1, and 0 otherwise. The party outputs 1 if at least $n - t - 1$ children have value 1, and 0 otherwise. The remaining difficulty is that an honest child may have already terminated and therefore sent no later acknowledgment. We next explain how the output function accounts for these missing messages.

Accounting for parties that have already terminated. The threshold argument above relies on honest children supporting the messages they receive. If an honest child has already terminated, however, it sends no further acknowledgments. Even a message from an honest party could then fall short of the required support. The output function accounts for this by filling certain empty entries, allowing a child to contribute support without requiring the corresponding party to send another message.

First consider a party that terminates in round $2 < t + 1$. Such an early termination requires an empty conflict graph. Consequently, the terminating party received a signed sender bit b, and every honest party's round-2 message contains that same signed bit. Although the terminating party sends no round-3 acknowledgment, we can still account for its support for these round-2 messages. Specifically, if a node's round-3 entry is empty but its round-2 entry contains a signed message giving the sender's bit b, we add V_b to the round-3 entry. A child with this value supports a parent's round-2 message containing the same signed sender bit b, but not the opposite bit. This preserves support for every honest party's round-2 message, since all these messages contain b. Giving the child unrestricted support would also allow it to support messages containing the opposite sender bit. The conflict test checks that a party stopping after round 2 could have satisfied this early-termination condition. If its round-2 message contains no signed sender bit, the observer adds a self-loop for that party. If the message contains a signed bit b, the observer adds a conflict between that party and any party whose round-2 message does not contain the same signed bit, including a party that sent no message.

For an empty entry in round $u \geq 4$, the output function instead uses $\star$. It does so only if an earlier entry at the same node contains either a signed round-2 message with a sender bit, or a signed message from a round between 3 and $u - 1$. An honest party that terminates early leaves such an earlier message. If it stops in round 2, its message contains a signed sender bit. If it stops later, it has already sent a round-3 message. When a child has $\star$ in its round-u entry, it counts toward the support for any signed round-$(u - 1)$ message considered at the parent. This does not claim that the child actually received the message. Rather, the rule allows the child to contribute support despite the missing acknowledgment. The candidate message must still appear in the party's collected information and have $n - t - d$ supporting children. Neither $\star$ nor V_b creates a signed message. Since $\star$ is used only in entries for rounds $u \geq 4$, it does not directly support round-2 messages. Those are supported by signed round-3 messages or by the bit-specific value V_b.

These rules preserve support for messages sent by honest parties, even after some parties terminate. We must still show that parties using different numbers of rounds compute the same

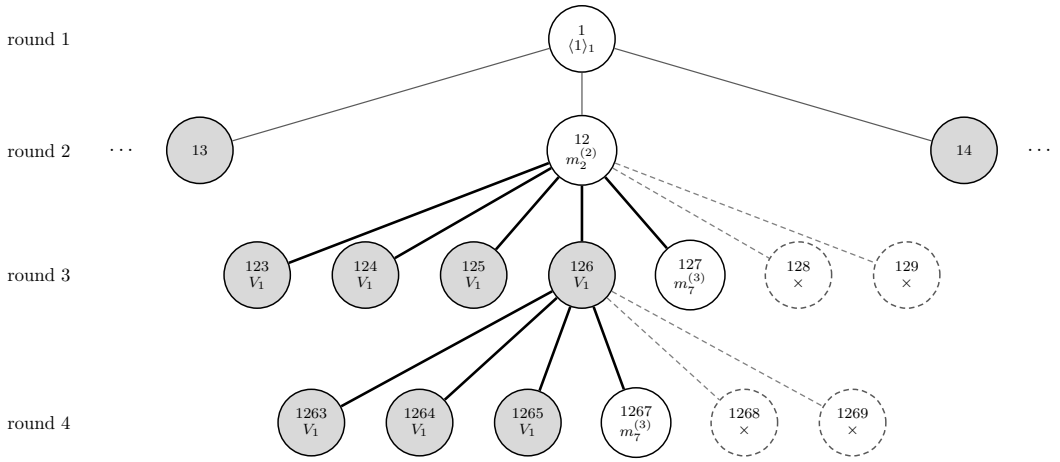

Figure 2: Nested support reconstruction in an EIG tree for $n = 9$ and $t = 3$. The sender P_1 is honest and signs $\langle 1 \rangle_1$. Parties P_3, P_4, P_5, P_6 terminate at the end of round 2, while P_7 continues through round 4 and executes $\Pi_{\mathsf{Output}}(7, 4)$. The second line of each node, when shown, indicates a signed message or a locally reconstructed entry; in particular, V_1 is not a message sent in that round. At $\widehat{126}$, the three children $\widehat{1263}$, $\widehat{1264}$, and $\widehat{1265}$ contribute V_1, while $\widehat{1267}$ contributes the signed round-3 report of P_7. These four children support the signed round-2 message of P_6, meeting the threshold $n - t - 3 = 3$. Since P_6 sent nothing in round 3, the resulting round-3 entry at $\widehat{126}$ is completed with V_1. The same computation gives V_1 at $\widehat{123}$, $\widehat{124}$, and $\widehat{125}$. These four completed entries, together with P_7's signed receipt list at $\widehat{127}$, support P_2's signed round-2 message $m_2^{(2)}$ at $\widehat{12}$, meeting the threshold $n - t - 2 = 4$. Shaded circles end at parties that terminated in round 2. Bold edges indicate support. Dashed circles end at faulty parties P_8, P_9, and dashed edges indicate lack of support. Only the subtree of $\widehat{126}$ is expanded to depth 4.

output. We explain this after describing how the output bit is chosen.

Why later information does not change the output. Consider the first honest party P_p to terminate in some round $3 \le r < t + 1$, and let S be a cover it accepts. Party P_p computes its output using messages through round $r - 1$. Before stopping, it also receives the round-r messages, which describe what other parties received in round $r - 1$. These messages allow it to check for inconsistencies in the last round of information used to compute its output. Suppose P_j and an honest party P_k both lie outside S. Through round $r - 1$, P_k must have received the same messages from P_j as P_p did, including in rounds when neither received a message. Otherwise, P_k's next signed message would give P_p the conflict $\{P_j, P_k\}$. Neither endpoint belongs to S, contradicting the fact that S covers every conflict. Likewise, any signed message from P_j that P_p has seen must match what it received directly in the corresponding round. A different signed message would create the self-loop $\{P_j, P_j\}$ and force P_j into S.

If the sender is honest, agreement follows trivially. Suppose, therefore, that the sender is faulty and belongs to S. Starting at the sender, follow a path through faulty parties in S until it reaches either an honest party or a party outside S. Since identities do not repeat and $|S| \le r - 2$, this happens at depth at most $r - 1$. If the path reaches an honest party, the output computation preserves that party's messages, giving the same entries at every honest party. Otherwise, it reaches a faulty party outside S. We distinguish whether this happens before depth $r - 1$ or exactly at depth $r - 1$. If it happens before depth $r - 1$, the node is internal both in P_p's computation and in every later output computation. The consistency checks and the support threshold guarantee that all honest parties compute the same entries there.

The remaining case is a path whose first $r - 2$ parties are faulty members of S, followed by a faulty party P_j outside S at depth $r - 1$. This node is a leaf in P_p's computation, so P_p uses the

messages it received directly from P_j. A party that stops later may instead process this node as an internal node and use its children to determine which messages to include. We must show that these two rules give the same result. The first $r-2$ parties on this path exhaust S, so every honest party is outside S. The conflict checks therefore ensure that every honest party received the same messages from P_j through round $r-1$. Each of these messages was acknowledged in the next round, and all honest parties remained active through round r. Consequently, the same messages that P_p uses directly at the leaf have enough honest children's support when the node is processed as an internal node later. A different message cannot obtain such support, because no honest child received it. If a message was missing, the honest parties agree on its absence and the earlier entries used by the completion rules. Thus, the entries agree whether the node is processed as a leaf or as an internal node. In both cases, the completion rules account for missing messages after honest parties terminate. Later output computations therefore give the same bit without requiring any further messages from P_p.

The construction achieves termination within $\min\{f+2, t+1\}$ rounds using a plain PKI. It still incurs exponential communication and local computation, since it does not reduce the number of EIG paths. Its early-stopping guarantee comes from checking whether the observed conflicts have a sufficiently small cover, while computing the output only when a party is ready to terminate.

1.3.3 An Early Stopping Broadcast Protocol when $t < n$

We now shift our focus to the dishonest majority setting, where the reasoning used in the $t < n/2$ regime fundamentally breaks down. The conceptual move from $t < n/2$ to $t < n$ is not a matter of tightening constants; it is a change in what can count as evidence. When $t < n/2$, many correctness arguments implicitly rely on the inequality $n - t > t$: if every party reports something, then even a fully coordinated adversary cannot outvote the honest parties, and "large" sets of reports necessarily contain substantial honest structure (e.g., honest majorities or honest intersections).

When $t < n$, these counting arguments become invalid. Honest majority may not exist, and in the worst case, the adversary can control almost all parties. In particular, any threshold that is $\leq t$ can be satisfied using only Byzantine signatures, and two honest parties can be shown disjoint "certificates" that share no honest witness. This forces protocols to pivot from reasoning about "votes" to reasoning about transferable evidence, and from "quorums" to "paths" or "histories": correctness must rely on what the adversary cannot fake (e.g., honest signatures) and on how such evidence propagates through the network.

A second issue arises from how information propagates. In the $t < n/2$ regime, once a value spreads sufficiently, the adversary cannot suppress it. This implicitly relies on the fact that any continuation of the execution must contain enough honest parties to carry the information forward. In the $t < n$ regime, such reasoning is no longer meaningful: there may exist arbitrarily long sequences of faulty parties, along which the adversary can delay, or selectively reveal information. Concretely, if a party does not receive a message in a given round, it only learns that the message was not delivered along that particular communication pattern. It does *not* rule out the possibility that the message is still propagating along a different unobserved sequence of parties. Thus, the absence of a message does not imply that the message will never be received, it only reflects that it has not been observed along any path explored so far. Previous solutions address this uncertainty using tools such as polarizers [26, 9], which construct forms of transferable evidence about missing messages. While computationally-efficient, these approaches rely on heavier agreement-style subroutines and incur additional rounds, preventing them from achieving optimal early-stopping guarantees.

These two issues together force a fundamental shift in perspective. Instead of relying on counting arguments ("enough votes"), we have to rely on information that the adversary cannot fake, and

reason about how this evidence can propagate through the network. We thus abandon the resolving rules of the $t < n/2$ protocol entirely, including its 1-biased signaling, which relies on silence as an implicit, unsigned vote. Path-based reasoning instead needs explicit signed evidence at every hop, something silence cannot provide.

Rather than asking whether a value has enough support, we must ask:

What evidence can still reach me in the future, and along which sequences of parties could it arrive?

To capture this, we adopt a different viewpoint based on *message paths*. Instead of reasoning about sets of messages, we reason about the different propagation histories through which the sender's value could still reach a party.

A message path is a sequence of distinct parties representing a potential chain of relays from the sender to a receiver. This perspective resolves the key difficulty: a party that has not yet received the sender's value cannot conclude that the value does not exist, as it may still be traveling along a longer path through intermediate parties. Therefore, the protocol tracks which such paths remain plausible, and a party terminates only once no plausible path remains along which the value could still arrive.

To understand this tool better, we can analyze the following scenario: Let P_1, P_2, P_3 be three parties and let $P_s = P_4$ be the sender with some input message x.

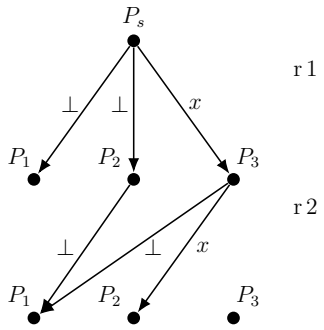

Figure 3: Example execution

We analyze the execution shown in Fig. 3. Note that the figure in round 2 only shows the messages that are relevant for the example, but we can assume that P_1 and P_2 send messages honestly in that round.

- In round 1, the sender P_s sends its input x only to party P_3.
- Only P_3 received x in round 1, and P_1, P_2 remain clueless about the fact that P_3 received x, but they do know that P_s is corrupt.
- In round 2, a corrupt P_3 could forward x to P_2, but send $\perp$ to P_1 to claim to not have received any message from the sender P_s.
- In round 2, P_1 observes both P_2 and P_3 to not have received a message. This might indicate to P_1 that no party observed the sender's input and therefore P_1 might terminate assuming the sender has not sent a message at all. This would however conflict with party P_2, that does observe the sender message (having received it from P_3).

To prevent P_1 from reaching the conclusion to terminate without having seen the sender's input, we can use message paths instead: In this scenario, P_1 initially considers all possible message paths of lengths 2, 3 and 4 (the different options on how the sender's input message might reach it): (P_s, P_1), (P_s, P_2, P_1), (P_s, P_3, P_1), (P_s, P_2, P_3, P_1), (P_s, P_3, P_2, P_1). Note that the length of a message path corresponds to the number of parties in it.

After round 1, P_1 can exclude the path (P_s, P_1) from the potential options, as not receiving a message in that round implies that the sender must be corrupted.

In round 2, P_1 can furthermore exclude the paths (P_s, P_2, P_1), (P_s, P_3, P_1), as both of them indicate no receipt of the sender's input. However, there are still paths (P_s, P_2, P_3, P_1), (P_s, P_3, P_2, P_1) of length 4 left. This would indicate to P_1 that terminating in round 2 could lead to an inconsistency among the honest parties.

This example demonstrates what a message path is, and how tracking which ones could still carry the sender's value tells a party when it is safe to stop. Note that this in itself is not enough: knowing that a path delivers a value from the sender does not yet say what that value is. We return to how a party reads off the value carried by a given path in Section 1.3.4; for now, it suffices to track which paths remain plausible.

Pruning Message Paths. Using message paths, a party maintains a set of paths that could still carry the sender's value, and gradually eliminates those that become inconsistent with what it has observed.

As an example: suppose P_2 does not receive a valid message from P_3 in some round, and reports this fact to P_1. Then P_1 knows that P_2 knows P_3 to be corrupted. P_2 can therefore disregard anything P_3 sends it from now on. As a result, P_1 can also safely disregard any message that would have to travel the path (P_3, P_2, P_1).

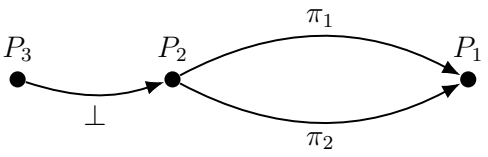

Figure 4: Observing party corruptions

This idea extends to any path, as shown in Fig. 4: if P_1 learns, via some message path π_1, rather than directly, that P_2 observed P_3 to be corrupted, the same reasoning applies. P_1 no longer needs to wait for any message that travels from P_3 to P_2, and then from P_2 along π_1 to P_1. Instead, it can safely remove every message path with the suffix (P_3, P_2, π_1, P_1), knowing that no further relevant information will follow there. Note, however, that P_1 cannot safely remove every message path containing the sequence (P_3, P_2) as a whole: there might still be an alternative longer path π_2, along which P_2 claims to have received a message from P_3 after all, reaching P_1 in a later round.

At round r, a party only needs to consider paths of length at least $r + 1$: shorter paths have already been evaluated, while longer ones may still carry the sender's value and must keep being tracked. Putting this together, a party starts with the set of all paths of length at most $t + 2$, prunes it round by round using the rule above, and evaluates what value each surviving path carries. Termination becomes possible exactly once no message paths remain.

Path Augmentations. Pruning alone is not sufficient for early termination: honest parties never observe one another behaving corruptly, so in an execution with no (or few) corruptions, the rule above never fires, and the path set never shrinks on its own. To address this, we introduce *path augmentations*, a second way to shrink the path set that does not rely on detecting corruption directly.

Suppose P_1 has confirmation that P_2 received the exact same message from P_3 that P_1 itself received directly. Then the longer path (P_3, P_2, P_1) cannot carry anything new, that P_1 does not already know from the shorter path (P_3, P_1): whatever value that longer path would deliver in the next round, P_1 already has it. P_1 can therefore treat (P_3, P_1) as standing in for (P_3, P_2, P_1) as well. We say that (P_3, P_2, P_1) is an augmentation of (P_3, P_1) and can stop tracking the longer path separately.

This idea generalizes further, as shown in Fig. 5. First, it composes directly with pruning: if P_1 observes, via some path (P_3, π, P_2, P_1), that P_3 marked P_4 as corrupted, it can remove every message path with the suffix $(P_4, P_3, \pi, P_2, P_1)$ exactly as in the pruning rule above. Augmentation extends this further: if some other party P' also confirms having received the same message from P_2 along that path, P_1 can additionally discard every path with the suf-

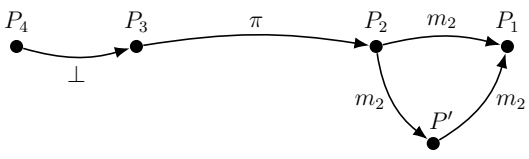

Figure 5: Using path augmentations for pruning

fix $(P_4, P_3, \pi, P_2, P', P_1)$, since P''s relay carries nothing new either. With a second confirming party P'', three more augmented suffixes become removable in the same way – one for P'' alone, and one for each ordering of P' and P'': $(P_4, P_3, \pi, P_2, P'', P_1)$, $(P_4, P_3, \pi, P_2, P', P'', P_1)$, and $(P_4, P_3, \pi, P_2, P'', P', P_1)$.

Combining pruning with augmentations dramatically reduces the space of possible histories. As the protocol progresses, all remaining paths become shorter. Once no plausible path remains through which new evidence could arrive, parties can safely terminate. This effectively collapses large classes of propagation histories into a smaller set of representative paths.

Realizing this requires more than P_1 simply trusting that P_2 confirmed the same message: for augmentation to be sound, this confirmation must be something any party can later verify, not just a private belief P_1 holds. We introduce an announcement mechanism to provide exactly this guarantee, built from a primitive we call *Confirming Weak Broadcast* (Appendix A): if two parties both confirm the same announcement from a third party, whatever they receive from this third party is guaranteed to match. It does not guarantee both parties receive the announcement at all, only that when they do, they agree.

Recall that pruning alone gets stuck when everyone behaves honestly, since no party observes an accusation. Path augmentation allows parties to remove longer paths once a shorter path has supplied the same information. Every path starting at the sender with length at least $3f+4$ contains three consecutive honest parties whose first party is not the original sender. Indeed, without such a triple, the initial honest run has length at most three and every subsequent honest run has length at most two, giving a total length of at most $3f + 3$. The first hop is left unchanged because the original sender does not announce its round-one message. The shortening argument therefore targets paths of length at most $3f + 3$, leading to a termination bound of $9f + 7$ communication rounds for $f \geq 1$.

1.3.4 Reading Message Paths: The Information Tree

It remains to explain how a party determines what a given message path actually carries: the value of the sender delivered along it, or, for pruning, whether some party along it reported having received nothing. For this, every party forwards in each round everything it has received so far, and stores what it receives in a tree, which we call the *Information Tree*. As in the EIG tree, nodes are sequences of parties and depth tracks the round, but the two trees are not identical. Both serve the same underlying purpose for the party that builds them: recording what every other party has reported having received, and through which sequence of relays. Where the two diverge is in what gets recorded at a node. The EIG tree stores, in each node, the value of the sender, observed over a sequence of parties. In the information tree, only the leaf-nodes (here at the top) store the value of the sender, and the path from the leaf to the root (at the bottom) corresponds to the sequence over which the party observes it. A message path is thus precisely a sequence of links from a leaf to the root, and a party reads the value carried by any path directly off its tree. The following figure shows an example of the EIG tree and information tree for $n = 4$ parties $\mathcal{P} = \{P_1, P_2, P_3, P_4 = P_s\}$ over 3 rounds, and shows how conflicting information is propagated throughout the tree.

In the EIG tree (left), a node is addressed by the full sequence of parties it passed through, read off the labels from the root down to that node: P_1's depth-2 nodes P_sP_2 and P_sP_3 record what P_2 and P_3, respectively, each reported to P_1 in round 2; its depth-3 nodes $P_sP_2P_3$ and $P_sP_3P_2$ record what P_1 heard, via one further relay, about what P_2 told P_3 and what P_3 told P_2.

To the right, we see that each node in the information tree links directly to the subtrees it received from every other party, rather than opening a new, uniquely-labelled child for every relay. Such a link carries no value of its own. The only thing it records is its *position*, i.e. which party it

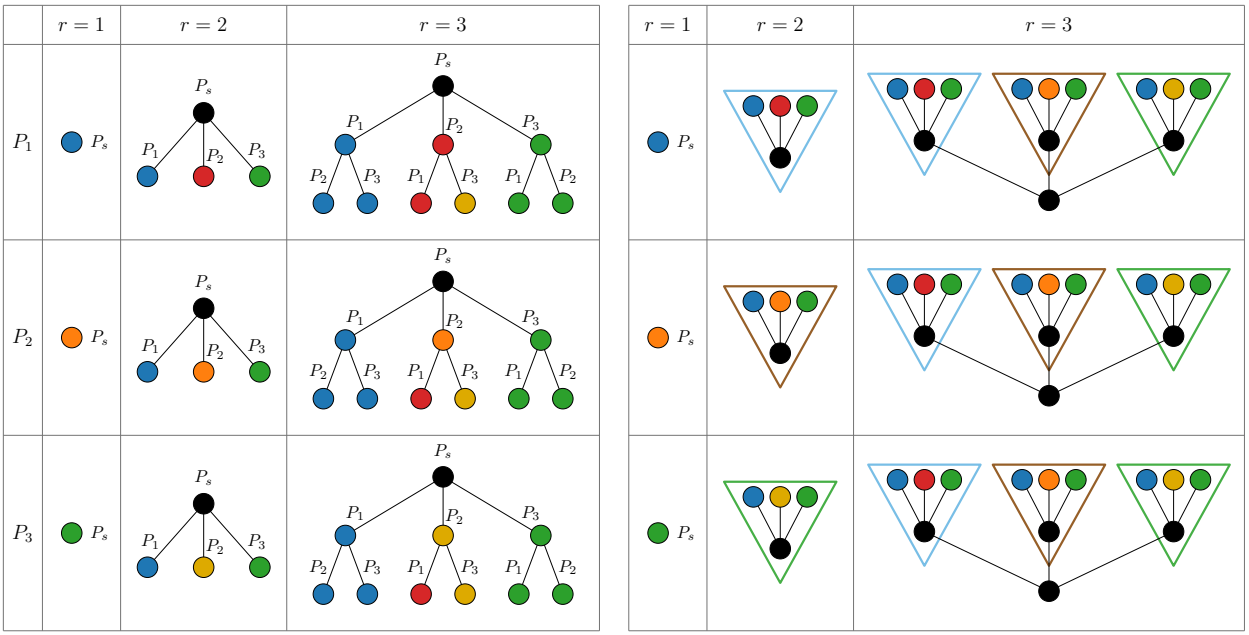

Figure 6: The classical EIG tree (left) versus the Information Tree (right), as built locally by each of three parties P_1, P_2, P_3 over three rounds. Each dot is colored by the party whose report it records: a **blue** dot is information recorded from P_1, a **green** dot from P_3, and an **orange** dot the sender's value P_2 received in the first round. Since an honest party reports the same thing to everyone, its dots keep a single color across every tree. The corrupt P_2 is the exception: its two conflicting reports are drawn **red** (the value P_2 sent toward P_1) and **yellow** (the value P_2 sent toward P_3).

came from. Reading off these positions along the sequence of connections from a leaf (at the top) down to the root recovers exactly the path along which the value of the sender was received. Each node after the first round could thus be labeled with the party it corresponds to. The one exception is round 1, where the label is the sender P_s, exactly as in the EIG tree's root.

The figure also shows how both representations deal with messages from a corrupted party. Suppose P_2 is corrupted and, in round 2, tells P_1 one value (red) and P_3 a different one (yellow). Both P_1 and P_3 learn of this by round 3, though the two trees make it visible in different ways. In the EIG tree, the mismatch sits within a single branch: P_1's depth-2 node P_sP_2 (its own direct record, red) and its child $P_sP_2P_3$ (P_3's relay of what P_2 told it, yellow) disagree, stacked directly below one another. In the information tree, the mismatch instead appears across sibling subtrees: within P_1's round-3 tree, the P_2-entry is red in P_1's own subtree but yellow in the subtree received from P_3. In terms of message paths, the paths (P_s, P_2, P_1) and (P_s, P_2, P_3, P_1) deliver different values.

1.4 Related Work

We introduce an overview of *deterministic* early-stopping broadcast protocols. Deterministic broadcast requires $\min\{f+2, t+1\}$ rounds of communication [24, 14, 13].

Protocols for $t < n/3$ in the Plain Model. The first early-stopping protocol with optimal resilience is due to Dolev, Reischuk, and Strong [12], and requires $\min\{2f+5, 2t+3\}$ rounds. The same work introduced the first protocol tightly matching the lower bound of $\min\{f+2, t+1\}$, but achieved a suboptimal resilience requiring $n > 2t^2 + 3t + 5$. Since then, there has been a significant

number of follow-up works [31, 33, 5, 13, 8, 21] focused on improving both axes, the concrete number of rounds and resilience.

The first protocol with optimal resilience and number of rounds was introduced in Berman, Garay and Perry [5], but required exponential message complexity. This was later improved by Abraham and Dolev, who introduced the first optimal early-stopping protocol with optimal resilience and polynomial message complexity [2].

Protocols for $t \geq n/3$ assuming a plain PKI for signatures. When $t \geq n/3$, it is known that some form of setup is needed [24], with a plain PKI for signatures being the most popular choice. In this setting, any number $t < n$ of corruptions can be tolerated [14], and much less work has been done on early-stopping protocols.

Honest-majority protocols. Perry and Toueg [29] introduced a protocol tolerating $t < n/2$ corruptions and running in $2f + 4$ rounds. This was improved much later by Elsheimy, Loss and Papamanthou [17] to $(1 + \epsilon)f + O(1)$ rounds for any constant $\epsilon > 0$. Their bound is $(d + 5) \cdot (\lfloor f/d \rfloor + 2) + 2$, where choosing a sufficiently large constant d gives the stated dependence on f. Elsheimy, Loss and Papamanthou subsequently improved this bound to $f + O(\sqrt{f})$ rounds with polynomial communication complexity [18]. The same work also gives an EIG protocol achieving decision within $\min\{f + 2, t + 1\}$ rounds, followed by an additional communication round before termination. Our protocol subsumes and strengthens their EIG result by achieving termination within $\min\{f + 2, t + 1\}$ rounds.

Subsequently, Choi et al. [7] also obtained optimal early-stopping broadcast for $t < n/2$ with exponential communication complexity. Unlike their approach, which resolves EIG nodes during execution, our construction uses conflict covers to determine when to terminate and computes the output only upon termination. We additionally obtain an asymptotically round-optimal early-stopping broadcast protocol tolerating any $t < n$ corruptions.

More recently, Elsheimy, Kamp, Loss and Papamanthou [16] achieved $f + O(\sqrt{f})$ rounds with quadratic communication complexity.

Dishonest-majority protocols. Loss and Nielsen [26] presented a broadcast protocol with optimal resilience of $t < n$ but suboptimal runtime of $O(\min\{f^2, t + 1\})$. Finally, in a recent work, Deligios, Klasovita and Liu-Zhang [9] introduced an asymptotically round-optimal protocol $O(\min\{f, t + 1\})$, but suboptimal resilience $t = (1 - \epsilon)n$.

Randomized Protocols. In the setting of randomized broadcast protocols where parties have access to local randomness, protocols can achieve an expected constant number of rounds in the honest majority setting, and can be made to achieve simultaneous termination after $O(\lambda)$ rounds with a negligible failure probability [19, 23, 27, 1, 20, 22]. Similarly, randomized protocols in the dishonest majority setting are known with an expected constant number of rounds [35] for $t = (1 - \varepsilon)n$ for constant ε, and fixed number of rounds $O(\lambda)$ with negligible probability of failure for $t = (1 - \varepsilon)n$. The works [34, 32, 3] achieved higher levels of adaptive resilience assuming time-lock assumptions.

Elsheimy, Loss and Papamanthou [17] also give a randomized early-stopping protocol for $t < n/2$ with constant expected round complexity and an $O(f + 1)$ worst-case round bound. Elsheimy, Kamp and Loss [15] use cryptographic sortition to achieve communication linear in n, up to factors in the security parameters, under weakly adaptive corruption. Their protocol tolerates $t < (1 - \epsilon)n/2$ corruptions for any constant $0 < \epsilon < 1$, has constant expected round complexity, and terminates in $O(f + 1)$ rounds with overwhelming probability.

2 Preliminaries

Network and Setup Assumptions. We assume 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 is 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 Vfy. 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 $\mathsf{Vfy}(\mathsf{pk}_i, \mathsf{sig}(m), m)$. We refer to a signature $\mathsf{sig}(m)$ as *valid* if $\mathsf{Vfy}(\mathsf{pk}_i, \mathsf{sig}(m), m) = 1$. 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.

Note on Signatures and EIG. For simplicity and in line with the overwhelming majority of the literature in this area, we model signature schemes as idealized primitives, i.e., with perfect correctness and unforgeability. We stress that some care must be taken when instantiating the idealized signature scheme with a computationally unforgeable one in the context of our early stopping EIG protocol, whose runtime may be exponential in n. Namely, for the protocol's security guarantees to hold, the signature's parameters must be set in such a way that the scheme remains unforgeable even when the adversary has running time exponential in n. While this is possible, it creates an implicit dependency between n and the size of a signature and comes at the cost of a steep efficiency penalty. However, we view both of these drawbacks acceptable in our context, as our early stopping EIG protocol is meant to be a feasibility result and already incurs exponential communication overhead (i.e., regardless of the size of signatures).

Adversary Model. We consider an adaptive, computationally-bounded Byzantine adversary that can corrupt up to $t < n/2$, resp. $t < n$, parties at any point of a protocol execution. A corrupt 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 any undelivered messages of a newly corrupted party P_i that P_i sent while it was still honest.
Finally, we give formal definition to the Byzantine Broadcast problem:.

Definition 3 (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 $y_i = v$;
- Agreement: All honest parties output the same value v.
- Termination: Every honest party terminates.

3 Early Stopping Broadcast in $\min\{f+2, t+1\}$

In this section, we introduce our early-stopping Byzantine binary broadcast protocol for $t < n/2$. Our construction follows the Exponential Information Gathering (EIG) paradigm: parties collect a complete record of how the sender's value has been reported through different sequences of parties and then use this information to compute their output. Unlike classical EIG protocols, however, a party need not wait until round $t+1$ to determine its output. Instead, it may terminate as soon as its observed inconsistencies can be explained by a sufficiently small set of potentially faulty parties.

At a high level, the protocol consists of two components. First, parties exchange signed reports of the messages they received in the preceding round. These reports are stored in an EIG tree and make it possible to compare what different parties claim to have received. Second, a party uses these comparisons to decide whether it has collected enough information to terminate. If so, it uses the information collected so far to compute the resolved value of the root of the EIG tree. The important distinction from the usual EIG approach is that we do not try to predict and close the value of each individual subtree during the communication phase. Instead, we use a separate early-termination test, while the output computation is performed only when the party terminates.

3.1 EIG Preliminaries

Exponential Information Gathering. Let Σ_d denote the set of all sequences of d distinct party identifiers that begin with s, where P_s is the designated sender. We define $\Sigma = \bigcup_{d=1}^{t+1} \Sigma_d$ and $\Sigma_1 = \{s\}$. An EIG tree is a data structure in which every node corresponds to a sequence in Σ, and a node is connected to the node representing its longest proper prefix. Since our protocol addresses the broadcast problem, the node $\hat{s}$ is the root of the tree. The root is at depth 1, and the depth of any other node is one greater than that of its parent. Thus, a node at depth d has $n-d$ children, corresponding to the parties that do not already appear in the sequence. In particular, the root has $n-1$ children.

We use w to denote a sequence of party identifiers and $\hat{w}$ to denote the corresponding node. The length of w, denoted $|w|$, equals the depth of $\hat{w}$. If $w = \alpha j$, then $\hat{\alpha}$ is the parent of $\hat{w}$, and we write $\widehat{wk}$ for a child of $\hat{w}$, where $P_k \notin w$. Each party P_i maintains its own EIG tree Tree_i. We write $\text{val}(\hat{w}, i)$ for the information stored by P_i at node $\hat{w}$. In particular, the root stores the sender's received signed round-1 message, or $\perp$ if no valid message was received. For $w = \alpha j$ of depth $r \geq 2$, the entry $\text{val}(\hat{w}, i)$ records the value at $\hat{\alpha}$ reported by P_j in round r, or $\perp$ if no valid report is available.

As in the standard EIG construction, a path represents a sequence along which information about the sender has been forwarded. For example, the node $\widehat{sj}$ records what P_j reported receiving from the sender, while $\widehat{sjk}$ records what P_k reported about P_j's earlier report. A malicious party may give different reports to different recipients, so the trees constructed by different honest parties need not be identical. Finally, because the number of possible paths grows rapidly with their length, the EIG representation incurs exponential communication and local computation in the worst case.

Signed reports and missing messages. Each party stores signed reports of received messages in its EIG tree. We use $m_{j\to i}^{(r)}$ to denote the valid signed message received directly by P_i from P_j in round r, and set $m_{j\to i}^{(r)} = \perp$ if no such message is received. Both sender values are transmitted as signed messages. Parties include their records of received and missing messages in subsequent reports, allowing recipients to detect discrepancies between their receipts.

3.2 Protocol Construction

The protocol follows a two-stage paradigm similar to the classical EIG solution. In the first stage, parties exchange signed messages for up to $t+1$ rounds, storing the information they receive in their local EIG trees. In the second stage, a party computes its output by applying a deterministic resolving function to this information. The difference is that a party may enter the second stage before the final round if the conflicts it has observed admit a small cover. We first explain the exchange of signed messages and the termination rule, before describing the output and cover subroutines in detail.

The designated sender P_s has initial input $v_s \in \{0,1\}$. In the first round, it signs v_s and sends $\langle v_s \rangle_s$ to every party. At the end of this round, each P_i stores the received signed message as $m^{(1)}_{s\to i}$, or stores $\perp$ if no valid message is received, and sets $\mathrm{val}(\hat{s}, i) := m^{(1)}_{s\to i}$. A faulty sender may sign conflicting values or withhold its message from some parties. Subsequent rounds record the resulting differences among the parties' receipts.

In every round $r = 2, \ldots, t+1$, each party P_i signs and broadcasts one message containing its complete list of receipts from the preceding round. More precisely, it sends $m^{(r)}_i := \left\langle r, \left(m^{(r-1)}_{j\to i}\right)_{P_j \in \mathcal{P}} \right\rangle_i$. Every entry is either the exact signed message that P_i received from the corresponding party in round $r-1$, or $\perp$ if no valid message was received.

At the end of round r, party P_i stores each valid signed message $m^{(r)}_{j\to i}$ it received, or $\perp$, and records its own sent message as $m^{(r)}_{i\to i} = m^{(r)}_i$. It then adds the depth-r nodes of the EIG tree. For every path $w = \alpha j$ with $|w| = r$, the value at $\hat{w}$ is the value at $\hat{\alpha}$ reported by P_j in its signed round-r message.

Determining when to output. At the end of every round $r \geq 2$, a party checks whether it should terminate. If $r = t+1$, the party computes its output using the entire tree and terminates. Otherwise, it invokes Π_{Cover} to determine whether its observed inconsistencies can be explained by at most $r-2$ parties. When this test succeeds, the party terminates. The sender always outputs its own input. A nonsender that terminates early in round 2 outputs the bit in the signed sender message received in round 1, or 0 if no valid sender message was received. If it terminates in a later round $r \leq t$, it instead outputs the value returned by $\Pi_{\mathsf{Output}}(i, r-1)$.

Protocol Π_{BB}

Assume sender P_s has input $v_s \in \{0,1\}$.

Initialization ($r = 0$):

1: P_i initializes Tree_i with root $\hat{s}$ and sets $\mathsf{Conflicts}_i := \varnothing$.

In round $r = 1$, every party P_i does the following:

1: **if** $P_i = P_s$ **then**
2: Sign and send $\langle v_s \rangle_s$ to all parties.

At the end of the round:
3: Store $m^{(1)}_{s\to i}$ and set $val(\hat{s}, i) := m^{(1)}_{s\to i}$.

In rounds $r = 2, \ldots, t+1$, every party P_i does:

1: Sign and send $m^{(r)}_i := \left\langle r, (m^{(r-1)}_{j\to i})_{P_j \in \mathcal{P}} \right\rangle_i$ to all parties.

At the end of the round:
2: Store the valid signed messages $m^{(r)}_{j\to i}$ (or $\perp$ if absent) and $m^{(r)}_{i\to i} := m^{(r)}_i$.
3: **for all** paths $w = \alpha j$ with $|w| = r$ **do**
4: **if** P_i received a valid message $m^{(r)}_j$ from P_j **then**

```
        Set val(ŵ, i) := val(α̂, j) as reported in m_j^(r).
    else
        Set val(ŵ, i) := ⊥.
if r = t + 1 then
    Output v_s if P_i = P_s, and Π_Output(i, r) otherwise.
    Terminate.
else if Π_Cover(i, r) = yes then
    if P_i = P_s then
        Output v_s.
    else if r = 2 then
        Output the bit in val(ŝ, i), or 0 if this entry is absent.
    else
        Output Π_Output(i, r − 1).
    Terminate.
```

3.3 Output Resolving Function

We next describe how a party calculates the value to output when it terminates. As in the classical EIG paradigm, the output is computed by processing the tree from the leaves toward the root. However, our resolving function does not apply a majority rule to a bit stored at every internal node. Instead, it first reconstructs the signed reports that are sufficiently supported by the descendants of each node, and then uses the resulting round-2 reports to determine the value of the root. This reconstruction is necessary because parties may terminate in different rounds. Consequently, the absence of a later signed report does not, by itself, imply that the reporting party behaved maliciously or that a previously observed value should be discarded.

Let P_i be an honest party executing $\Pi_{\mathsf{Output}}(i, h)$. The subroutine processes the EIG tree from depth h toward the root. For each node $\hat{w}$ of depth d ending at P_j, and each $2 \le u \le d$, it computes $\mathrm{val}^{(u)}(\hat{w}, i)$, a set of signed round-$u$ messages of P_j retained by the output computation. Unlike the entries $\mathrm{val}(\hat{w}, i)$ recorded during the execution, these sets are computed only when P_i terminates. A set may contain multiple messages if a faulty P_j signed conflicting reports that each satisfy the support threshold. When no signed message is retained, the completion rules may instead assign a special value to account for missing reports.

Reconstructing messages. Leaf nodes are reconstructed to the messages received during protocol execution before termination. Now consider a node $\hat{w}$ of depth $d < h$ that ends at P_j. The party P_i considers each signed round-u message $m_j^{(u)}$ of P_j that appears in the EIG information available through round h. Rather than accepting such a message merely because it was observed along one path, P_i checks whether the children of $\hat{w}$ provide sufficient support for it. A child $\widehat{wk}$ can support $m_j^{(u)}$ if its reconstructed round-$(u+1)$ entry contains a signed message from P_k explicitly reporting receipt of $m_j^{(u)}$ in round u. A missing signed message can be completed from earlier reports. The message $m_j^{(u)}$ is added to $\mathrm{val}^{(u)}(\hat{w}, i)$ if at least $n - t - d$ distinct children support it. The threshold $n - t - d$ serves two purposes. A node at depth d has $n - d$ possible children, of which at least $n - t - d$ are honest. Thus, when P_j is honest and sends a signed message in round u, the reports of its honest children, together with the completion rules below for children that have already terminated, preserve this message in the reconstruction. Second, the threshold prevents a message at an all-faulty path from being accepted solely on the reports of the remaining faulty parties. If the d parties on the path are faulty, there are at most $t - d$ faulty children, whereas

$n - t - d > t - d$ holds because $n > 2t$. Thus, in this case an accepted message must have support involving an honest child.

Completing missing entries. The remaining difficulty is that an honest party may terminate before the end of the execution and therefore have no signed messages in subsequent rounds. For this reason, the output function introduces the completion values V_0, V_1, and $\star$. These values are computed locally by the output function and do not represent signed messages sent by the corresponding party.

Consider an honest party that terminates at the end of round 2 after reporting the sender's bit $b \in \{0, 1\}$ in a signed message. To account for the party's missing round-3 message, we introduce the completion value V_b, which records the bit b reported in round 2. Specifically, if no signed round-3 message has been stored at a node $\hat{w}$, but its round-2 entry contains a signed message of the party reporting the sender's bit b, the protocol adds V_b to the round-3 entry. A child whose round-3 entry contains V_b supports a signed round-2 message reporting $\langle b \rangle_s$, but does not support a round-2 message reporting the opposite bit.

For a missing entry in a round $u \geq 4$, the protocol uses the marker $\star$ to account for a party that stopped after previously sending a valid report. It assigns $\star$ provided that an earlier entry contains either a signed round-2 report of a sender value or a signed message from some round $3 \leq v < u$. The marker $\star$ can support candidate signed messages in the reconstruction at earlier nodes.

Resolving the root. After reconstructing the signed messages, the party assigns a bit to each child $\widehat{sj}$ of the root. It sets $\mathrm{val}^\star(\widehat{sj}, i) = 1$ if $\mathrm{val}^{(2)}(\widehat{sj}, i)$ contains a signed round-2 message from P_j reporting $\langle 1 \rangle_s$, and sets the value to 0 otherwise. The root is then resolved to 1 if at least $n - t - 1$ children have value 1, and to 0 otherwise:

$$\mathrm{val}^\star(\hat{s}, i) = 1 \quad \Longleftrightarrow \quad \left|\{j \neq s : \mathrm{val}^\star(\widehat{sj}, i) = 1\}\right| \geq n - t - 1.$$

The subroutine returns this resolved value of the root. It also returns 0 immediately if P_i has not observed a valid sender signature on 1 by round h, since in that case no supported report in this output computation can justify an output of 1.

The threshold $n - t - 1$ differs from the majority threshold used in the classical EIG construction. To understand why we choose this threshold, first consider the case where the sender is honest. If the sender has input 1, every honest nonsender reports $\langle 1 \rangle_s$ in round 2. Since there are at least $n - t - 1$ such parties, the output function assigns the root value 1. Conversely, if the sender has input 0, no valid signed message $\langle 1 \rangle_s$ can exist, and the output function returns 0.

Now, suppose an honest party terminates as early as round 2 with output b. In this case, its conflict set must be empty, which implies that every honest party reported the same signed sender value $\langle b \rangle_s$ in round 2. If $b = 1$, there are enough honest parties reporting $\langle 1 \rangle_s$ to satisfy the threshold $n - t - 1$, and hence the output function returns 1. On the other hand, if $b = 0$, no honest party reports $\langle 1 \rangle_s$. If the sender is honest, it never signed a message on 1, so the output must be 0. Otherwise, the sender is faulty, and at most $t - 1$ faulty nonsenders could report $\langle 1 \rangle_s$. Since $n - t - 1 \geq t$, these parties alone cannot satisfy the threshold required to assign the root value 1. Therefore, the output function is consistent with the decision of an honest party that terminates in round 2, even when other honest parties continue executing the protocol.

Protocol Π_{Output}

Execution:

1: **if** P_i has not seen a signed $\langle 1 \rangle_s$ by the end of round r **then**
2: **return** 0.

3: **for** $d = r, r-1, \ldots, 2$ **do** // From leaves to the root
4: **for all** paths $w = \alpha j$ with $|w| = d$ **do**
5: **for** $u = 2, \ldots, d$ **do**
6: **if** $P_j = P_i$ **then**
7: Set $val^{(u)}(\hat{w}, i) := \{m_i^{(u)}\}$, where $m_i^{(u)}$ is P_i's own signed message sent in round u.
8: **else if** $d = r$ **then** // Leaf node
9: **if** P_i received a valid signed message $m_j^{(u)}$ directly from P_j in round u **then**
10: Set $val^{(u)}(\hat{w}, i) := \{m_j^{(u)}\}$.
11: **else**
12: Set $val^{(u)}(\hat{w}, i) := \varnothing$.
13: **else**// Non-leaf node
14: Set $val^{(u)}(\hat{w}, i) := \varnothing$.
15: **for all** signed round-u messages $m_j^{(u)}$ by P_j appearing in Tree_i by round r **do**
16: **if** at least $n - t - d$ distinct children $\widehat{wk}$, $P_k \notin w$, support $m_j^{(u)}$ **then**
17: Add $m_j^{(u)}$ to $val^{(u)}(\hat{w}, i)$.

Complete empty entries:

18: **if** $val^{(u)}(\hat{w}, i) = \varnothing$ **then**
19: **if** $u = 3$ **then**
20: **for all** $b \in \{0, 1\}$ **do**
21: **if** $val^{(2)}(\hat{w}, i)$ contains a signed message reporting $\langle b \rangle_s$ **then**
22: Add V_b to $val^{(3)}(\hat{w}, i)$.
23: **else if** $u \geq 4$ **then**
24: **if** $val^{(2)}(\hat{w}, i)$ contains a signed message reporting $\langle 0 \rangle_s$ or $\langle 1 \rangle_s$, or some $val^{(v)}(\hat{w}, i)$, $3 \leq v < u$, contains a signed message **then**
25: Set $val^{(u)}(\hat{w}, i) := \star$.

Resolve the children of the root:

26: **for all** $P_j \in \mathcal{P} \setminus \{P_s\}$ **do**
27: **if** $val^{(2)}(\widehat{sj}, i)$ contains a signed round-2 message reporting $\langle 1 \rangle_s$ **then**
28: Set $val^{\star}(\widehat{sj}, i) := 1$.
29: **else**
30: Set $val^{\star}(\widehat{sj}, i) := 0$.

Resolve the root:

31: **if** $\left|\{j \neq s : val^{\star}(\widehat{sj}, i) = 1\}\right| \geq n - t - 1$ **then**
32: Set $val^{\star}(\hat{s}, i) := 1$.
33: **else**
34: Set $val^{\star}(\hat{s}, i) := 0$.
35: **return** $val^{\star}(\hat{s}, i)$.

Child support. A child $\widehat{wk}$ supports a signed round-u message $m_j^{(u)}$ from P_j if one of the following holds:

1. $val^{(u+1)}(\widehat{wk}, i)$ contains a signed round-$(u+1)$ message from P_k reporting that it received $m_j^{(u)}$ from P_j in round u.
2. $u \geq 3$ and $val^{(u+1)}(\widehat{wk}, i) = \star$.
3. $u = 2$, $m_j^{(2)}$ reports $\langle b \rangle_s$, and $V_b \in val^{(3)}(\widehat{wk}, i)$ for some $b \in \{0, 1\}$.

3.4 Early Termination via Conflict Covers

We now explain the early-termination rule. In the standard EIG protocol, parties gather information through round $t+1$ because a faulty party may continue revealing different reports along previously unexplored paths. Our construction avoids waiting for the full tree whenever the observed inconsistencies can already be attributed to a small set of potentially faulty parties. Importantly, parties need not identify exactly which parties are corrupted, nor must all honest parties agree on the same set of suspected parties. Instead, each party records constraints saying that at least one of two parties must be faulty and checks whether a small set intersects every such constraint.

Conflict pairs. Each party P_i maintains a set Conflicts_i of unordered pairs $\{P_j, P_k\}$. The meaning of a pair is that P_j and P_k cannot both be honest in an execution consistent with the evidence observed by P_i. We also use pairs of the form $\{P_j, P_j\}$, which certify that P_j itself must be faulty. The conflict set is initially empty and is updated as new signed reports are received. Protocol Π_{Cover} adds pairs according to four checks, which we describe next.

Checking the sending sequence. First, P_i verifies that a party's valid messages follow the sending sequence required by the protocol. Every honest party sends in round 2, and an honest party that terminates stops sending permanently. Therefore, if P_i does not receive a valid round-2 message from P_j, or receives a new valid message from P_j after a round in which its message was missing, P_i adds $\{P_j, P_j\}$ to its conflict set. For the designated sender, the absence of a valid round-1 message is treated in the same way.

Checking conflicting signed messages. Second, P_i examines the signed messages that appear anywhere in its EIG tree. If it finds a signed round-u message attributed to P_j that differs from the message it received directly from P_j in round u, it adds $\{P_j, P_j\}$. For example, a malicious P_j might send one signed receipt list to P_i and a different signed list to another party, which subsequently forwards it. Since an honest party signs at most one message in a given round and sends that same message to all parties, the two conflicting signed lists prove that P_j is faulty. The same reasoning applies when P_i received no valid message directly from P_j but later observes a signed message from P_j for that round inside another report.

Checking reported receipts. Third, P_i compares the receipts claimed by other parties with the messages it received directly. Suppose the signed round-u message of P_k reports receiving some message from P_j in round $u-1$, but P_i's own receipt from P_j in that round is different, including the case where one of the receipts is $\perp$. Then P_i adds $\{P_j, P_k\}$ to its conflict set. This discrepancy does not, by itself, prove that P_j is faulty, as P_j might have sent different messages to P_i and P_k, or P_k might have lied about its receipt. However, if both P_j and P_k were honest, they would have produced matching receipts.

Parties that stop after round 2. The fourth check deals specifically with a party whose valid sending sequence ends in round 2. Such a party might be an honest party that terminated as early as possible. An honest party can terminate in round 2 only when its conflict set is empty, so it must have received a signed sender value $\langle b \rangle_s$ in round 1 and observed no disagreement with this value among the round-2 reports of honest parties. Accordingly, if P_j stops after round 2 but its round-2 message does not report a signed sender value, P_i adds $\{P_j, P_j\}$. Otherwise, if P_j's round-2 message reports $\langle b \rangle_s$ and some party P_k's round-2 message does not contain this same signed value (including when P_k sent nothing), P_i adds $\{P_j, P_k\}$.

Finding a small cover. After applying the four checks, P_i tests whether there exists a set $S \subseteq \mathcal{P} \setminus \{P_i\}$ with $|S| \leq r-2$ such that $S \cap \{P_j, P_k\} \neq \varnothing$ for every $\{P_j, P_k\} \in \text{Conflicts}_i$. In other words, S is a vertex cover of the graph represented by the conflict pairs. Π_{Cover} returns yes if such a

set exists and no otherwise. The protocol does not require the party to find the smallest possible cover, only to determine whether one of size at most $r-2$ exists.

The key property is that every conflict detected by an honest party has at least one faulty endpoint. Let F be the set of parties corrupted at any point during the execution, with $|F| = f$. Then F covers every pair in $\mathsf{Conflicts}_i$ for every honest P_i. Consequently, if a party has not terminated earlier, the cover test must succeed by round $f+2$, because at that point $|F| = f = (f+2)-2$. A party that does not terminate through the cover test terminates at the latest in round $t+1$. Thus, the termination rule yields the desired bound $\min\{f+2, t+1\}$.

Protocol Π_{Cover}

$\mathsf{Conflicts}_i$ contains pairs $\{P_j, P_k\}$, indicating that at least one of P_j and P_k is faulty. A pair $\{P_j, P_j\}$ indicates that P_j is faulty.

Execution:

Check the sending sequence:
for all $P_j \in \mathcal{P}$ **do**
  if P_i did not receive a valid message from P_j in round 2, or did not receive the sender's round-1 message when $P_j = P_s$, or received a message from P_j after a missing message **then**
    Add $\{P_j, P_j\}$ to $\mathsf{Conflicts}_i$.

Check conflicting signed messages:
for all signed round-u messages $m_j^{(u)}$ from P_j appearing in Tree_i **do**
  if $m_j^{(u)}$ differs from the message P_i received directly from P_j in round u **then**
    Add $\{P_j, P_j\}$ to $\mathsf{Conflicts}_i$.

Check reported messages:
for all signed round-u messages $m_k^{(u)}$ from P_k appearing in Tree_i, with $u \geq 2$ **do**
  for all $P_j \in \mathcal{P}$ **do**
    if P_k reports receiving a different message from P_j in round $u-1$ than the message received directly by P_i from P_j in that round **then**
      Add $\{P_j, P_k\}$ to $\mathsf{Conflicts}_i$.

Check parties that stopped in round 2:
for all P_j whose valid sending sequence ends in round $2 < r$ **do**
  if P_j's round-2 message does not report a signed sender's value **then**
    Add $\{P_j, P_j\}$ to $\mathsf{Conflicts}_i$.
  else
    for all $P_k \in \mathcal{P}$ such that $m_{k\to i}^{(2)}$ does not contain the sender's signed bit reported in $m_{j\to i}^{(2)}$, including $m_{k\to i}^{(2)} = \perp$ **do**
      Add $\{P_j, P_k\}$ to $\mathsf{Conflicts}_i$.

Termination condition:
if there exists a set $S \subseteq \mathcal{P} \setminus \{P_i\}$ with $|S| \leq r-2$ such that every pair in $\mathsf{Conflicts}_i$ contains at least one party in S **then**
  return yes.
else
  return no.

4 Proof of Correctness of Π_{BB}

Lemma 4 (Honest Forwarding)**.** *Let P_i and P_j be honest parties, and let w be an EIG path such that $P_j \notin w$. If both parties are active in round $|w|+1$, then $val(\widehat{wj}, i) = val(\hat{w}, j)$.*

Proof. In round $|w|+1$, honest party P_j sends its information from the previous round to all parties.

In particular, its message reports $val(\hat{w}, j)$. Since P_i and P_j are honest and active, P_i receives this message from P_j. By the EIG update rule, P_i sets $val(\widehat{wj}, i) = val(\hat{w}, j)$.

□

Lemma 5. *Let F be the set of parties corrupted during the execution. For every honest party P_i and every pair $\{P_j, P_k\} \in \mathsf{Conflicts}_i$, at least one of P_j and P_k belongs to F.*

Proof. We show that none of the four checks in Π_{Cover} can add a pair consisting entirely of honest parties.
Sending-prefix check. An honest party sends a valid message in every required round until it terminates. After termination, it never sends again. Therefore, an honest party cannot violate the sending-prefix condition.
Conflicting signed messages. An honest party signs exactly one message in each round and sends the same message to all honest parties. Since signatures cannot be forged, every valid signed message attributed to an honest party P_j agrees with the message received directly from P_j. Thus, this check cannot add $\{P_j, P_j\}$ for an honest P_j.
Reported-receipt check. Suppose P_j and P_k are honest. Whenever P_k sends a message in round u, it reports exactly what it received from P_j in round $u-1$. Since P_i and P_k are both honest and active in round $u-1$, they receive the same message from P_j, or both receive nothing if P_j has already terminated. Therefore, their receipts agree, and the check cannot add $\{P_j, P_k\}$.
Round-2 stopping check. Suppose an honest party P_j terminates at the end of round 2 and another honest party remains active in a later round. Since the protocol has not reached its final round, P_j must have terminated because Π_{Cover} returned yes. In round 2, the termination condition requires a covering set of size at most 0. Consequently, $\mathsf{Conflicts}_j$ must be empty. In particular, P_j must have received a valid signed source message in round 1. Otherwise, the sending-prefix check would have added $\{P_s, P_s\}$. Furthermore, every honest party P_k must have reported the same signed source value as P_j in round 2. Otherwise, the reported-receipt check at P_j would have added $\{P_s, P_k\}$, contradicting $\mathsf{Conflicts}_j = \varnothing$. Therefore, the round-2 stopping check cannot add $\{P_j, P_j\}$ or $\{P_j, P_k\}$ when both parties are honest. Hence, every pair in $\mathsf{Conflicts}_i$ contains at least one faulty party. □

Lemma 6 (Early Termination)**.** *Let F be the set of parties corrupted during the execution, and let $f = |F| \le t$. Every honest party terminates by the end of round* $\min\{f+2, t+1\}$.

Proof. Fix an honest party P_i. By the protocol, every party that has not terminated earlier terminates at the end of round $t+1$. It therefore suffices to consider the case $f+2 < t+1$. Suppose P_i has not terminated before round $f+2$. By Lemma 5, every pair in $\mathsf{Conflicts}_i$ contains at least one party in F. Thus, F covers every pair in $\mathsf{Conflicts}_i$. Since P_i is honest, $P_i \notin F$. Moreover, $|F| = f = (f+2) - 2$. Hence, F satisfies the termination condition of Π_{Cover} at the end of round $f+2$. Consequently, Π_{Cover} returns yes, and P_i outputs and terminates in round $f+2$. Therefore, every honest party terminates by the end of round $\min\{f+2, t+1\}$. □

Lemma 7 (Round-2 Reports)**.** *If an honest party terminates in round $2 < t+1$ with output b, then every honest party reports $\langle b \rangle_s$ in its round-2 message.*

Proof. Let P_i be an honest party that terminates in round 2 with output b. Since P_i terminates before round $t+1$, Π_{Cover} must return yes. In round 2, the termination condition requires a set of at most $2-2=0$ parties covering every conflict. Thus, $\mathsf{Conflicts}_i = \varnothing$. First, P_i must have received a valid $\langle b \rangle_s$ from the sender in round 1. Otherwise, the sending-prefix check would have added $\{P_s, P_s\}$ to $\mathsf{Conflicts}_i$, contradicting that the set is empty. Now consider any honest party P_j. Since no honest party terminates before round 2, P_j sends its round-2 message to all parties, including P_i.

Suppose P_j reports a sender's value different from $\langle b \rangle_s$, or reports no sender's value. Then P_j's round-2 message disagrees with the message that P_i received directly from P_s in round 1. By the reported-receipt check, P_i adds $\{P_s, P_j\}$ to $\mathsf{Conflicts}_i$. This contradicts $\mathsf{Conflicts}_i = \varnothing$. Therefore, every honest party reports $\langle b \rangle_s$ in round 2. □

Lemma 8 (Entries at Honest Leaf Nodes)**.** *Let P_i be an honest party executing $\Pi_{\mathsf{Output}}(i, h)$, and let w be a leaf ending at an honest party P_j. For every $2 \le u \le h$, the following holds:*

1. *If P_j sends $m_j^{(u)}$ in round u, then $val^{(u)}(\hat{w}, i) = \{m_j^{(u)}\}$.*
2. *If P_j terminates in round 2 with output b, then $val^{(3)}(\hat{w}, i) = \{V_b\}$.*
3. *If P_j terminates before round $u \ge 4$, then $val^{(u)}(\hat{w}, i) = \star$.*

Proof. Suppose P_j sends $m_j^{(u)}$ in round u. Since P_j is honest, its signed message is sent to P_i. By the output rules, $val^{(u)}(\hat{w}, i) = \{m_j^{(u)}\}$. Now suppose P_j terminates before round u. Since P_j sends no messages after termination, $val^{(u)}(\hat{w}, i)$ is initially empty. If $u = 3$, then P_j terminated in round 2 with output b. Its signed round-2 message reports $\langle b \rangle_s$. Therefore, the completion rule sets $val^{(3)}(\hat{w}, i) = \{V_b\}$.

If $u \ge 4$, then either P_j terminated in round 2, or it remained active through round 3. In the first case, its signed round-2 message reports a signed source value. In the second case, its signed round-3 message is available. Thus, in both cases, the completion condition is satisfied, and $val^{(u)}(\hat{w}, i) = \star$. □

Lemma 9 (Entries of Honest Parties)**.** *Let P_i be an honest party executing $\Pi_{\mathsf{Output}}(i, h)$, and let w be an EIG path of length $d \le h$ ending at an honest party P_j. For every $2 \le u \le d$,*

1. *If P_j sends $m_j^{(u)}$ in round u, then $val^{(u)}(\hat{w}, i) = \{m_j^{(u)}\}$.*
2. *If P_j terminates in round 2 with output b, then $val^{(3)}(\hat{w}, i) = \{V_b\}$.*
3. *If P_j terminates before round $u \ge 4$, then $val^{(u)}(\hat{w}, i) = \star$.*

Proof. We prove the lemma by induction from the leaves toward the root. The base case follows from Lemma 8.

Consider an internal node $\hat{w}$ ending at an honest party P_j. Assume the lemma holds for all children of $\hat{w}$.

Messages sent by P_j. Suppose P_j sends $m_j^{(u)}$ in round u. We show that every honest child $P_k \notin w$ supports this message. If P_k sends in round $u+1$, its message reports receiving $m_j^{(u)}$ from P_j. By the induction hypothesis, this report appears in $val^{(u+1)}(\widehat{wk}, i)$ and supports $m_j^{(u)}$. Otherwise, P_k has already terminated. If $u = 2$, then P_k terminated in round 2 with output b. By Lemma 7, P_j also reported $\langle b \rangle_s$. By the induction hypothesis, the child's entry contains V_b, which supports $m_j^{(2)}$. If $u \ge 3$, the induction hypothesis gives $val^{(u+1)}(\widehat{wk}, i) = \star$, which also supports $m_j^{(u)}$. Thus, every honest child outside w supports $m_j^{(u)}$. Since $|w| = d$, there are at least $n - f - d \ge n - t - d$ such children. Therefore, $m_j^{(u)}$ satisfies the support threshold. Since P_j is honest, P_i received this message directly, and no other signed round-u message from P_j exists. Hence, $val^{(u)}(\hat{w}, i) = \{m_j^{(u)}\}$.

Messages after termination. Suppose P_j has terminated before round u. Since P_j is honest, no signed round-u message from P_j exists. Thus, the entry is initially empty. If $u = 3$, then P_j

terminated in round 2 with output b. By the first part of the proof, its signed round-2 message is included in $val^{(2)}(\hat{w}, i)$. Since this message reports $\langle b \rangle_s$, the completion rule sets $val^{(3)}(\hat{w}, i) = \{V_b\}$. If $u \geq 4$, then either P_j terminated in round 2, or it sent a message in round 3. In both cases, the corresponding signed message is included in the earlier entry, as shown above. Therefore, the completion rule sets $val^{(u)}(\hat{w}, i) = \star$.

□

Lemma 10 (Validity)**.** *If the sender is honest with input b, every honest party outputs b.*

Proof. Since the sender P_s is honest, it sends $\langle b \rangle_s$ to all parties in round 1. Therefore, every honest party receives $\langle b \rangle_s$. If P_i terminates in round 2, it outputs the bit received directly from the sender, which is b. Otherwise, P_i executes Π_{Output} when it terminates. We show that this computation also returns b.
Case $b = 0$. Since the sender is honest and has input 0, it never signs $\langle 1 \rangle_s$. Moreover, no faulty party can forge the sender's signature. Thus, P_i cannot receive a valid $\langle 1 \rangle_s$. By the first step of Π_{Output}, P_i returns 0.
Case $b = 1$. Every honest party $P_j \neq P_s$ receives $\langle 1 \rangle_s$ in round 1 and reports this value in its signed round-2 message $m_j^{(2)}$. By Lemma 9, $m_j^{(2)}$ is included in $val^{(2)}(\widehat{sj}, i)$. Therefore, according to the output rule, P_i sets $val^{\star}(\widehat{sj}, i) = 1$ for every honest party $P_j \neq P_s$. Since the sender is honest, there are $n - f - 1$ honest parties other than the sender. Moreover, $f \leq t$ implies $n - f - 1 \geq n - t - 1$.

Hence, at least $n - t - 1$ children of the root are assigned value 1. By the final rule of Π_{Output}, P_i sets $val^{\star}(\hat{s}, i) = 1$ and outputs 1. Thus, in both cases, every honest party outputs the sender's input b. □

Lemma 11 (Agreement after round-2 termination)**.** *If an honest party terminates in round $2 < t+1$ with output b, then every honest party outputs b.*

Proof. If the sender is honest, agreement follows from Lemma 10. Hence, assume the sender is faulty. Let P_i be an honest party that terminates in round 2 with output b. By Lemma 7, every honest party P_j reports $\langle b \rangle_s$ in its signed round-2 message. Consider another honest party P_k. If P_k also terminates in round 2, it outputs the bit received directly from the sender in round 1, which is b. Otherwise, P_k terminates in a later round and executes Π_{Output}. We show that it also outputs b.
Case $b = 1$. Every honest party $P_j \neq P_s$ sends a signed round-2 message reporting $\langle 1 \rangle_s$. By Lemma 9, this message is included in $val^{(2)}(\widehat{sj}, k)$ when P_k executes Π_{Output}. Therefore, P_k sets $val^{\star}(\widehat{sj}, k) = 1$ for every honest party $P_j \neq P_s$. Since the sender is faulty, there are $n - f$ honest parties, all of which are nonsenders. Moreover, $f \leq t$ implies $n - f \geq n - t - 1$. Thus, at least $n - t - 1$ children of the root are assigned value 1. By the output rule, P_k sets $val^{\star}(\hat{s}, k) = 1$ and outputs 1.
Case $b = 0$. Every honest party P_j sends a signed round-2 message reporting $\langle 0 \rangle_s$. By Lemma 9, these messages are included in the corresponding entries of Π_{Output}. Since honest parties report $\langle 0 \rangle_s$, none of their round-2 messages can cause a child of the root to be assigned value 1. Consequently, only a faulty party P_j can have $val^{\star}(\widehat{sj}, k) = 1$. Since the sender is faulty, at most $f - 1$ nonsender parties are faulty. Therefore, at most $f - 1 \leq t - 1$ children of the root can be assigned value 1. Thus, fewer than $n - t - 1$ children can be assigned value 1. The output threshold is not satisfied, so P_k sets $val^{\star}(\hat{s}, k) = 0$ and outputs 0. Therefore, in both cases, every honest party that terminates after round 2 outputs b. Since all honest parties terminating in round 2 also output b, agreement follows.

□

Lemma 12 (Honest witness)**.** *Let w be a path of length $d < h$ containing only faulty parties. Suppose all honest parties are active in round $u+1$. If a signed message $m_j^{(u)}$ is included in $val^{(u)}(\hat{w}, i)$, then some honest party P_k reports receiving $m_j^{(u)}$ in its signed round-$(u+1)$ message. Consequently, every honest party receives $m_j^{(u)}$ by the end of round $u+1$.*

Proof. Since w contains d faulty parties, at most $f-d$ faulty parties remain outside w. For $m_j^{(u)}$ to be included in $val^{(u)}(\hat{w}, i)$, it must be supported by at least $n-t-d$ distinct children. Since $n > 2t$ and $f \leq t$, we have $n-t-d > f-d$. Therefore, at least one supporting child P_k must be honest.

By assumption, P_k sends a message $m_k^{(u+1)}$ in round $u+1$. By Lemma 9, its entry in the output computation contains exactly this signed message. Since P_k supports $m_j^{(u)}$, the child-support rule implies that $m_k^{(u+1)}$ reports receiving $m_j^{(u)}$ from P_j in round u. Finally, since P_k is honest, it sends $m_k^{(u+1)}$ to every honest party. Thus, every honest party receives $m_j^{(u)}$ within this message by the end of round $u+1$. □

Lemma 13 (Equal final-round entries)**.** *If no honest party terminates before round $t+1$, then every honest party computes identical entries $val^{(u)}(\hat{w}, i)$ for every EIG path w consisting only of faulty parties.*

Proof. We prove the claim by induction on decreasing path length, considering rounds u in increasing order at each node.
Base case: $d = f$. Consider a path w of length f containing only faulty parties. Since w contains all f faulty parties, every child $\widehat{wk}$ ends at an honest party P_k. By Lemma 9, every honest party computes the same entry at each child $\widehat{wk}$. Therefore, every honest party counts the same number of children supporting any given signed message.
Inductive step: $d < f$. Consider a path w of length d containing only faulty parties. Assume the claim holds for all longer paths containing only faulty parties. Consider any child $\widehat{wk}$ of $\hat{w}$. If P_k is honest, then by Lemma 9, every honest party computes the same entry at $\widehat{wk}$. Otherwise, P_k is faulty, so wk is a longer path containing only faulty parties. By the induction hypothesis, every honest party also computes the same entry at $\widehat{wk}$. Thus, every child of $\hat{w}$ has the same computed entry at all honest parties. Consequently, every honest party counts the same number of children supporting each signed message. It remains to show that every signed message included by one honest party is also known to all other honest parties. Suppose an honest party P_i includes a signed message $m_j^{(u)}$ in $val^{(u)}(\hat{w}, i)$. By Lemma 12, some honest party P_k reports receiving $m_j^{(u)}$ in its signed round-$(u+1)$ message. Since P_k is honest, it sends this message to every honest party. Therefore, every honest party receives a copy of $m_j^{(u)}$ by the end of round $u+1$.

Hence, every honest party considers the same signed messages and counts the same number of supporting children for each message. Therefore, every honest party includes exactly the same signed messages in $val^{(u)}(\hat{w}, i)$. Finally, if an entry is empty, the completion rules determine its value using earlier entries at the same node. Since these earlier entries agree at all honest parties, the completion rules also produce the same value. □

Lemma 14 (Agreement at the final round)**.** *If no honest party terminates before round $t+1$, then all honest parties output the same bit.*

Proof. If the sender is honest, it follows from validity. Otherwise, each root child ends either at an honest party, where all entries agree by Lemma 9, or at a faulty party, where the path contains only faulty parties and Lemma 13 applies. All parties therefore assign the same bit to each root child and use the same root threshold. □

We now consider the case where honest parties terminate in different rounds. Let r, where $3 \leq r < t+1$, be the first round in which an honest party terminates. Let P_p be an honest party that terminates in round r. Since P_p terminates in round r, it computes its output using messages through round $r-1$. Other honest parties may terminate later and use additional messages. We show that these additional messages cannot change the output.

Let S be a set accepted by Π_{Cover} at P_p in round r, where $|S| \leq r-2$. Every conflict detected by P_p contains at least one party in S. Since no honest party terminates before round r, all honest parties send messages through round r.

We first establish the consequences of the cover checks for parties outside S, and use them to show that the entries agree at paths ending at such parties. We then handle paths of length $r-1$, whose nodes are leaves in P_p's output computation but may be internal nodes in a later computation. Together with the entries of honest parties, these results allow us to propagate agreement toward the root when the sender belongs to S. When the sender is outside S, the cover checks instead ensure that sufficiently many round-2 reports agree with the sender value received by P_p.

Lemma 15 (No Gaps in Received Messages)**.** *Let S be a cover accepted by P_p in round r. For every $P_j \notin S$, the following holds: P_p receives a valid round-2 message from P_j. If P_p receives no message from P_j in a later round, it receives no further messages from P_j through round r.*

Proof. Consider a party $P_j \notin S$. Suppose first that P_p does not receive a valid round-2 message from P_j. By the sending-sequence check, P_p adds $\{P_j, P_j\}$ to $\mathsf{Conflicts}_p$. Now suppose P_p receives no message from P_j in some round, but receives another valid message from P_j in a later round. The sending-sequence check also adds $\{P_j, P_j\}$ in this case. However, since S is accepted by Π_{Cover}, every conflict must contain at least one party in S. The conflict $\{P_j, P_j\}$ can only be covered if $P_j \in S$, contradicting our assumption. Therefore, neither situation can occur, which proves the lemma. □

Lemma 16 (Consistency of Signed Messages)**.** *Let S be a cover accepted by P_p in round r. For every $P_j \notin S$, all signed round-u messages of P_j observed by P_p agree with the message it received directly from P_j in round u.*

Proof. Consider a party $P_j \notin S$. Suppose P_p observes a signed round-u message from P_j that differs from the message it received directly from P_j in that round. For example, P_p might receive one message directly from P_j and later observe a different signed message from P_j forwarded by another party. The same applies if P_p received no message directly from P_j but later observes a signed message from P_j for that round.

In either case, the signed-message check adds $\{P_j, P_j\}$ to $\mathsf{Conflicts}_p$. Since S covers every conflict, this would require $P_j \in S$, contradicting $P_j \notin S$. Therefore, every signed message of P_j observed by P_p agrees with the message received directly from P_j in the corresponding round. □

Lemma 17 (Receipts Outside the Cover)**.** *Let $P_j, P_k \notin S$, where P_k is honest. For every $2 \leq u \leq r-1$, parties P_p and P_k receive the same round-u message from P_j, or both receive no message.*

Proof. Fix a round $2 \leq u \leq r-1$. If $P_k = P_p$, the claim is immediate. Otherwise, suppose that P_p and P_k receive different messages from P_j in round u, including the possibility that one receives no message. Since r is the first round in which an honest party terminates, P_k is still active in round $u+1$. As P_k is honest, its round-$(u+1)$ message reports exactly what it received from P_j in round u. Moreover, P_k sends this message to all parties, including P_p.

When P_p receives this message, it compares the round-u message reported by P_k with the message it received directly from P_j in that round. By our assumption, these two messages differ.

Therefore, the reported-receipt check of Π_{Cover} adds $\{P_j, P_k\}$ to $\mathsf{Conflicts}_p$. However, S is a cover accepted by P_p. Hence, every conflict in $\mathsf{Conflicts}_p$ must contain at least one party in S. Since $P_j, P_k \notin S$, the conflict $\{P_j, P_k\}$ cannot be covered by S, a contradiction. Therefore, P_p and P_k receive the same message from P_j in round u. □

Lemma 18. *Suppose $P_j \notin S$ and P_p receives no messages from P_j after round 2. Then P_j's round-2 message reports some signed value $\langle b \rangle_s$, and every $P_k \notin S$ reports the same signed value in round 2.*

Proof. By the sending-sequence lemma, P_p received a valid round-2 message from P_j. Suppose this message does not report a signed sender value. By the round-2 stopping check, P_p adds $\{P_j, P_j\}$ to $\mathsf{Conflicts}_p$. This contradicts $P_j \notin S$, since S must cover every conflict. Therefore, P_j reports some $\langle b \rangle_s$ in round 2. Now suppose another party $P_k \notin S$ reports a different sender's value, or no sender value, in round 2. The round-2 stopping check then adds $\{P_j, P_k\}$ to $\mathsf{Conflicts}_p$. Since neither P_j nor P_k belongs to S, this conflict cannot be covered by S, a contradiction. Hence, every party outside S reports the same signed sender's value. □

Lemma 19. *Let w be a path of length $d \le r-2$ containing only faulty parties, and let $P_j \notin S$ be the last party on w. Suppose an honest party P_i executes $\Pi_{\mathsf{Output}}(i,h)$ with $h \ge r-1$. If a signed round-u message $m_j^{(u)}$ is included in $val^{(u)}(\hat{w}, i)$, then $m_j^{(u)} = m_{j\to p}^{(u)}$.*

Proof. Since $d \le r-2 < h$, the node $\hat{w}$ is an internal node in the output computation. Suppose $m_j^{(u)}$ is included in $val^{(u)}(\hat{w}, i)$. By Lemma 12, some honest party P_k reports receiving $m_j^{(u)}$ in its signed round-$(u+1)$ message. Since $u \le d \le r-2$, we have $u+1 \le r-1$. Moreover, no honest party terminates before round r. Therefore, P_k sends its round-$(u+1)$ message to P_p, which receives it before terminating. Since this message contains $m_j^{(u)}$, P_p also observes the same signed round-u message from P_j. By Lemma 16, every signed message from P_j observed by P_p must agree with the message P_p received directly from P_j in that round, since $P_j \notin S$. Consequently, $m_j^{(u)} = m_{j\to p}^{(u)}$. □

Lemma 20. *Let w be a path of length $2 \le d \le r-2$ containing only faulty parties, and let $P_j \notin S$ be the last party on w. Every honest party executing $\Pi_{\mathsf{Output}}(i,h)$, with $r-1 \le h \le t+1$, computes the same entries $val^{(u)}(\hat{w}, i)$ for $2 \le u \le d$.*

Proof. We prove the stronger claim that $val^{(u)}(\hat{w}, i) = \{m_{j\to p}^{(u)}\}$ whenever P_p received a message from P_j in round u, and otherwise the entry equals $\{V_b\}$ for $u = 3$ or $\star$ for $u \ge 4$, where b is the signed sender's bit reported in P_j's round-2 message. These are the only cases because every party outside S sends a valid round-2 message to P_p. We use induction on decreasing path length, considering rounds u in increasing order at each node.

Base case: $d = r-2$. Consider a path w ending at $P_j \notin S$. Suppose P_p received a signed message $m_{j\to p}^{(u)}$ in round u. By Lemma 17, every honest party $P_k \notin S$ received the same message from P_j in round u. Since no honest party terminates before round r, each such party reports receiving $m_{j\to p}^{(u)}$ in round $u+1$. By Lemma 9, these reports are included in the corresponding child entries and therefore support $m_{j\to p}^{(u)}$. There are at least $n - f - |S|$ honest parties outside S. Since $f \le t$ and $|S| \le r-2 = d$, we have $n - f - |S| \ge n - t - d$. Thus, the message satisfies the support threshold and is included in $val^{(u)}(\hat{w}, i)$ at every honest party. By Lemma 19, no different signed round-u message from P_j can be included. Therefore, all honest parties compute the same entry.

Inductive step: $d < r-2$. Assume the claim holds for all longer paths ending at faulty parties outside S. Consider a path w of length d ending at $P_j \notin S$, and suppose P_p received $m_{j\to p}^{(u)}$ in round u. We show that every child $\widehat{wk}$ with $P_k \notin S$ and $P_k \notin w$ supports this message. If P_k is honest, the

claim follows from Lemmas 17 and 9, as in the base case. Otherwise, P_k is faulty. By the induction hypothesis, its entry is determined by the messages received directly by P_p. If this entry contains a signed message, the reported-receipt check guarantees that it reports $m^{(u)}_{j\to p}$. If the entry contains V_b, the round-2 stopping check guarantees that $m^{(2)}_{j\to p}$ reports $\langle b\rangle_s$. Finally, if the entry is $\star$, it supports the message by the output rule. Thus, every child outside $S \cup w$ supports $m^{(u)}_{j\to p}$. There are at least $n - |S| - d$ such children. Since $|S| \le r - 2 \le t$, we have $n - |S| - d \ge n - t - d$. Therefore, the message satisfies the support threshold at every honest party. It is also known to every honest party through the reports of honest children. By Lemma 19, no different signed message can be included. Hence, all honest parties compute the same signed-message entry.

It remains to consider the case where P_p receives no message from P_j in round u. By Lemma 19, no signed round-u message from P_j can be included in the output computation. By Lemma 15, P_p received a valid round-2 message from P_j. Hence, $u \ge 3$. If $u = 3$, then P_j sent a message in round 2 but not in round 3. By Lemma 18, its round-2 message reports $\langle b\rangle_s$ for some bit b. Since this message is included in $val^{(2)}(\hat{w}, i)$, the completion rule assigns V_b. If $u \ge 4$, then either P_j stopped sending after round 2, or it sent a message in round 3. In the first case, its round-2 message reports a signed sender value. In the second case, its round-3 message is included in the earlier entry. Thus, the completion rule assigns $\star$ in both cases. Since the completion rules use only earlier entries at the same node, which agree by induction on u, all honest parties compute the same values for the missing entries. This completes the induction. □

Lemma 21 (Uniform receipts at the boundary)**.** *Let w contain only faulty parties, have length $r - 1$, and end at $P_j \notin S$. Suppose its proper prefix consists of parties in S. Every honest party received exactly $m^{(u)}_{j\to p}$ from P_j for $2 \le u \le r - 1$.*

Proof. The proper prefix has $r - 2$ distinct faulty identities and therefore exhausts S. Every honest party is outside S. Apply Lemma 17 for each u. For $u = r - 1$, the relevant evidence is the honest parties' round-r reports, which P_p received before it stopped. □

Lemma 22 (The Leaf-to-Internal Boundary)**.** *Let w contain only faulty parties, have length $r - 1$, and end at $P_j \notin S$. Suppose its proper prefix consists of parties in S. Then, every honest party P_i executing $\Pi_{\mathsf{Output}}(i, h)$, with $h \ge r - 1$, computes the same entries $val^{(u)}(\hat{w}, i)$ for all $2 \le u \le r - 1$.*

Proof. First, suppose P_i executes Π_{Output} using messages through round $r - 1$. Since $|w| = r - 1$, the node $\hat{w}$ is a leaf. By the leaf rule, its entries are determined by the messages P_i received directly from P_j. By Lemma 21, all honest parties received the same messages from P_j. Therefore, they compute the same entries at $\hat{w}$. Now suppose P_i executes Π_{Output} using messages through a later round $h > r - 1$. In this case, $\hat{w}$ is an internal node. Suppose P_p received a signed message $m^{(u)}_{j\to p}$ from P_j in round u. By Lemma 21, every honest party P_k received the same message from P_j in round u. Since $u + 1 \le r$, every honest P_k reports this message in round $u + 1$. By Lemma 9, these reports appear in the corresponding child entries and support $m^{(u)}_{j\to p}$.

Since w contains only faulty parties, all $n - f$ honest parties are children of $\hat{w}$. Moreover, $n - f \ge n - t - (r - 1)$. Thus, $m^{(u)}_{j\to p}$ satisfies the support threshold and is included in $val^{(u)}(\hat{w}, i)$. By Lemma 12, any different signed message would also require an honest supporting child. However, all honest parties received the same message from P_j in round u. Therefore, no different signed message can be included. Finally, if P_p received no message from P_j in round u, no signed round-u message can satisfy the support threshold. The entry is therefore empty before completion. Since all honest parties have the same earlier entries at $\hat{w}$, the completion rules assign the same V_b or

$\star$. Hence, the entries computed at $\hat{w}$ are identical whether the node is evaluated as a leaf or as an internal node. □

Lemma 23 (Agreement When the Sender Is in the Cover)**.** *Suppose $P_s \in S$. Every honest party P_i executing $\Pi_{\mathsf{Output}}(i, h)$ using messages through round $h \geq r - 1$ computes the same output bit.*

Proof. We show that all honest parties compute the same entries at the children of the root. Consider paths starting at the sender and consisting only of faulty parties in S. We first consider the children of these paths that end at either an honest party or a faulty party outside S. If a path ends at an honest party, its entries are identical at all honest parties by Lemma 9. If a path ends at a faulty party outside S, its entries are identical by Lemma 20 when its length is at most $r - 2$, and by Lemma 22 when its length is $r - 1$.

Since $|S| \leq r - 2$, these cases cover every path that leaves S. We now prove that the entries also agree at paths consisting only of faulty parties in S, by induction from the deepest such paths toward the root. Consider a path w of length $d \leq |S|$. Since $d \leq r - 2 < h$, the node $\hat{w}$ is an internal node in every output computation. By the induction hypothesis, every honest party computes the same entries at all children of $\hat{w}$. Now suppose a signed message $m_j^{(u)}$ is included in $val^{(u)}(\hat{w}, i)$ by some honest party P_i. By Lemma 12, some honest party P_k reports receiving $m_j^{(u)}$ in its signed round-$(u + 1)$ message. Since $u \leq d \leq r - 2$, we have $u + 1 \leq r - 1$. Thus, every honest party receives this signed report before round r, and therefore knows $m_j^{(u)}$.

Moreover, since the children's entries are identical at all honest parties, every honest party counts the same number of children supporting $m_j^{(u)}$. Hence, $m_j^{(u)}$ satisfies the support threshold at every honest party. The same argument applies to every signed message included by any honest party. Therefore, all honest parties include exactly the same signed messages in $val^{(u)}(\hat{w}, i)$. If an entry is empty, the completion rules determine its value using earlier entries at the same node. Since these entries also agree, every honest party obtains the same completed value. Consequently, all honest parties compute the same entries at every child of the root. They therefore assign the same bit to each root child and obtain the same final output using the threshold $n - t - 1$. □

Lemma 24 (Agreement When the Sender Is Outside the Cover)**.** *If $P_s \notin S$, then every honest party outputs the bit b that P_p received from the sender in round 1.*

Proof. If the sender is honest, the claim follows from Lemma 10. Hence, assume the sender is faulty. Since $P_s \notin S$, the sending-sequence check guarantees that P_p received a valid signed message $\langle b \rangle_s$ in round 1. Otherwise, it would add $\{P_s, P_s\}$, which cannot be covered by S. Moreover, every party $P_j \notin S$ reports $\langle b \rangle_s$ in its round-2 message received by P_p. Otherwise, the reported-receipt check would add $\{P_s, P_j\}$, which cannot be covered by S.

Case $r \geq 4$. Consider any nonsender $P_j \notin S$. If P_j is honest, its signed round-2 message is preserved by Lemma 9. If P_j is faulty, the path sj contains only faulty parties and has length $2 \leq r - 2$. By Lemma 20, its round-2 message is included in $val^{(2)}(\widehat{sj}, i)$ in every honest output computation using messages through round $h \geq r - 1$. Thus, every party outside S contributes the bit b to the final output computation. If $b = 1$, at least $n - 1 - |S| \geq n - t - 1$ children of the root are assigned 1. Therefore, every honest party outputs 1. If $b = 0$, only parties in S can contribute 1. Since $|S| \leq r - 2 < t \leq n - t - 1$, there are not enough such parties to satisfy the output threshold. Hence, every honest party outputs 0.

Case $r = 3$. Since $|S| \leq 1$, there are at least $n - f - |S| \geq n - t - 1$ honest parties outside S. All these parties report $\langle b \rangle_s$ in round 2, and their messages are preserved by Lemma 9. Therefore, if $b = 1$, at least $n - t - 1$ children of the root are assigned 1, and every honest party outputs 1. Now

suppose $b = 0$. We show that no honest party can report $\langle 1 \rangle_s$ in round 2. Otherwise, its signed round-2 message would contain $\langle 1 \rangle_s$. Since every honest party sends in round 2, P_p would receive this message. The signed-message check would then detect that the sender signed both $\langle 0 \rangle_s$ and $\langle 1 \rangle_s$, adding $\{P_s, P_s\}$ to $\mathsf{Conflicts}_p$. This contradicts $P_s \notin S$. Therefore, only faulty parties can contribute 1 to the output computation. There are at most $f - 1 \leq t - 1$ such parties, which is fewer than the threshold $n - t - 1 \geq t$. Hence, every honest party outputs 0. In both cases, every honest party outputs b. □

Theorem 25 (Agreement)**.** *All honest parties output the same bit.*

Proof. If the sender is honest, agreement follows from Lemma 10. Hence, suppose the sender is faulty.

First, suppose an honest party terminates in round $2 < t + 1$. By Lemma 11, every honest party outputs the same bit. Next, suppose the first honest party terminates in round $3 \leq r < t + 1$. Let P_p be an honest party terminating in round r, and let S be a cover accepted by P_p, with $|S| \leq r - 2$. If $P_s \in S$, agreement follows from Lemma 23. Otherwise, $P_s \notin S$, and agreement follows from Lemma 24.

Finally, suppose no honest party terminates before round $t+1$. Then all honest parties terminate in the final round, and agreement follows from Lemma 14. Therefore, all honest parties output the same bit. □

Theorem 26 (Correctness)**.** Π_{BB} *satisfies validity, agreement, and termination, and every honest party halts by the end of round* $\min\{f + 2, t + 1\}$.

Proof. Follows from Lemmas 10 and 6 with Theorem 25. □

5 On the Way to Achieve Optimal Resilience: Base Broadcast Protocol without Early Termination

As discussed in Section 1.3, the protocol of Section 3.2 relies on the fact that, for $t < n/2$, any sufficiently large set of reports must contain an honest majority. Once $t \geq n/2$, this counting argument breaks down: a coordinated adversary can produce a set of signed reports of size $\geq n/2$ without a single honest party among the signers. We address this by shifting from reasoning about *how many* parties report a value to reasoning about *which sequences of parties* such a report could have travelled through: a *message path*, as introduced in Section 1.3. A party no longer checks, if enough parties vouched for some value x. Instead, it verifies if there is a plausible chain of parties, consistent with everything observed up to that point, along which x could reach them. This section formalizes exactly that idea and proves it correct.

The protocol we present here, Π_{base}, achieves broadcast (Definition 3) for any $t < n$, but runs for $t + 2$ rounds, and incurs exponential communication and computation complexity. This might seem unnecessary and impractical, since there are already protocols without early termination that run in $t + 1$ rounds [14] that achieve polynomial communication complexity. However, this protocol serves as a base for the next section. We will show how the same message-path approach, augmented with an announcement mechanism, yields a protocol that terminates within $O(f)$ rounds whenever f parties are actually malicious.

Concretely, each round of Π_{base} follows the same pattern:

1. Every party forwards to all other parties the valid information it accumulated in the previous round.

2. Based on the information gathered so far, a party checks whether it can safely terminate.

3. Every party receives one message from each other party. A missing, malformed, or otherwise invalid message (e.g. carrying an invalid signature) is treated identically to receiving $\perp$.

4. The newly validated messages are used to update the current state of the given party.

The last step is what distinguishes this approach from non-EIG protocols: a party does not just fold new messages into a compact summary, but forwards, next round, precisely the (nested) messages it used to update its state.

The remainder of this section is organized as follows. Section 5.1 introduces the protocol-specific notation used to track and prune message paths; Section 5.2 presents Π_{base} itself together with the round-by-round intuition behind it; and Section 5.3 proves that Π_{base} achieves validity, agreement, and termination.

5.1 Protocol-Specific Preliminaries

Simplifying Assumptions. Without loss of generality, we fix a few conventions that keep the presentation clean. We assume that a corrupted party never sends multiple messages, or a message of the wrong format (like sending a message with an inconsistent round number), within a single round. Since an honest party never does so either, observing this behavior is itself direct evidence of corruption. A party that does misbehave in such a way can therefore be treated by every recipient as having sent nothing at all ($\perp$) in that round.

Messages and the Information Vector. Every message sent in Π_{base} (beyond the sender's very first one) carries a full summary of everything its author has received so far, addressed by the party it concerns rather than by the path it travelled along – the concrete realization of the *Information Tree* introduced in Section 1.3.4. As discussed in that section, a party needs to keep track of all message paths and evaluate which ones could still carry the sender's value to it. We store all this information in the *information vector* I, a nested data structure that mirrors the information tree and lets a party determine what value is propagated along any given message path, rather than only along one chain fixed in advance.

Formally, every message has the form (r, I, σ), where r is a round number, σ a signature over I, and I is the *information vector*. It is an array, indexed by parties, summarizing everything its author has observed up to round r. We write $\mathsf{I}_i^{(r)}$ for the information vector held by party P_i at the end of round r:

- If $r = 1$, $\mathsf{I}_i^{(1)}$ has a single entry, for the sender: $\mathsf{I}_i^{(1)}[P_s] = m$, where m is either the sender's signed input $(0, x, \sigma)$, or $\perp$ if no valid message from the sender was received.

- If $r > 1$, $\mathsf{I}_i^{(r)}$ has one entry for every party $P_j \in \mathcal{P} \setminus \{P_s\}$: $\mathsf{I}_i^{(r)}[P_j]$ is the message P_i received from P_j in round r, or $\perp$ if none (or an invalid one) was received. We occasionally refer to $\mathsf{I}_i^{(r)}[P_j] = \perp$ as "P_i accusing P_j."

Since P_j's round-r message is itself of the form $(r-1, \mathsf{I}_j^{(r-1)}, \sigma_j)$, an information vector recursively contains the entire information vectors of every party who, directly or indirectly, forwarded something to P_i.

Validity of Messages. Because I nests the party's entire transcript, checking that a received message is valid means recursively re-verifying every signature nested inside it, not just the outermost

one. A plain EIG-style check of the outermost signature alone would not suffice, since it says nothing about whether the rest of the reported history is itself coherent. Protocol `checkValidity` makes this precise. This is what makes messages *transferable* between honest parties: whether a message is valid depends only on public information (signatures, round numbers), so if one honest party accepts a message, any other honest party who later receives that very same message will accept it too.

Protocol `checkValidity`(r, m_k)

A party checks the message m_k received from party P_k in round r:

If $r = 1$, let $m_s = (0, x, \sigma_s)$ (then P_k is the sender P_s):

1: Return $\mathsf{Vfy}_{\mathsf{pk}_s}(x, \sigma_s)$

Else, let $m_k = (r-1, \mathsf{I}_k, \sigma_k)$

1: If not $\mathsf{Vfy}_{\mathsf{pk}_k}(\mathsf{I}_k, \sigma_k)$: return `false`
2: If $r = 2$: return `true` if $\mathsf{I}_k[P_s] = \perp$, else return `checkValidity`$(r-1, \mathsf{I}_k[P_s])$
3: For each party $P_j \in \mathcal{P} \setminus \{P_s\}$ s.t. $\mathsf{I}_k[P_j] \neq \perp$:
4: If not `checkValidity`$(r-1, \mathsf{I}_k[P_j])$: return `false`
Return `true`

The special case for $r = 2$ above exists because the information vector constructed in round 1 (and sent in round 2) has only a single entry (the sender's message) rather than the general shape of entries from all parties.

Message Paths. Recall from Section 1.3 that a *message path* $\pi = (P_1, \ldots, P_j)$ denotes one possible chain of relays (P_1 to P_2, P_2 to P_3, and so on) along which the sender's value could reach a party, and we refer to the number of parties in π as its *length*. We extend our array-access notation so that, given a nested information vector, we can directly read off the value or message that is propagated along a given path. We write $\mathsf{I}_i[(P_k, P_j)]$ for the message P_j claims to have received from P_k, as reported towards P_i; more generally, $\mathsf{I}_i[\pi]$ for a path $\pi = (P_1, \ldots, P_j)$ denotes the message that travelled $P_1 \to P_2 \to \cdots \to P_j \to P_i$.

We allow appending the receiving party itself to a path, i.e. treating $\mathsf{I}_i[(P_1, \ldots, P_j)]$ and $\mathsf{I}_i[(P_1, \ldots, P_j, P_i)]$ interchangeably: this is useful, as we will use message paths to track the flow of information from the sender to a given party, and it is convenient to have the path end at that party itself (rather than at the last hop before it). Protocol $\mathsf{I}[\pi]$ gives the formal (recursive) definition.

Protocol $\mathsf{I}_i[\pi]$

If $\pi = (P_j)$.

1: Return $\mathsf{I}_i[P_j]$

Else let $\pi = (P_1, P_2, \ldots, P_k, P_l)$.

2: If $P_l = P_i$:
3: Return $\mathsf{I}_i[(P_1, P_2, \ldots, P_k)]$
4: Else:
5: Let $\mathsf{I}_i[P_l] = m_l = (r, \mathsf{I}_l, \sigma_l)$
6: Return $\mathsf{I}_l[(P_1, P_2, \ldots, P_k)]$

If π begins with the sender P_s, $\mathsf{I}_i[\pi]$ resolves down to the sender's original round-1 message, $(0, x, \sigma)$, or $\perp$. We write $\mathsf{I}_i[\pi].\mathsf{val}$ for the value x it carries: $(0, x, \sigma).\mathsf{val} := x$, and $\perp.\mathsf{val} := \perp$.
Each party P_i maintains a set M_i of message paths (from the sender to itself) that it still considers *plausible* to carry the sender's value. Initially, M_i contains every such path of length at most $t + 2$. A path of length $t + 2$ will contain another honest party besides P_i itself, making any longer paths

redundant to track.

Throughout protocol execution, P_i removes paths from M_i as soon as it observes evidence that the paths will not carry the sender's value to it. We say that $M_i^{(r)}$ denotes this set at the end of round r.

Observed Values. Finally, P_i maintains a set V_i of values it has observed to be signed by the sender. This is a set rather than a single candidate because a corrupted sender may sign several different, conflicting values to different parties; observing $|V_i^{(r)}| > 1$ is itself conclusive evidence that the sender is corrupted. As with M_i, we write $V_i^{(r)}$ for the set at the end of round r.

5.2 The Protocol

We can now present Π_{base} itself. In every round, a party first sends the valid information it accumulated in the previous round to all parties, and checks whether it may already terminate; if not, it receives the incoming messages, checks them for validity, and uses them to update the local information M_i and V_i.

To be able to precisely argue about the set M_i in any specific round (and throughout the execution), we will use $M_{init-i}^{(r)}$ to denote the set at the beginning of round r, and $M_i^{(r)}$ to be the set at the end of the round. Similarly to the sets M_i, we will use $V_{init-i}^{(r)}$ to denote the set at the beginning of round r, and $V_i^{(r)}$ to be the set at the end of the round. This will help us to formally argue about the information P_i observes at any point in time.

In round 1, the sender P_s signs and sends its input, and every other party P_i expects to see this single, correctly-signed message. If it does, x is added to V_i; otherwise the sender sent nothing, a badly-signed message, or multiple conflicting ones. In that case P_i sets $\mathsf{I}_i^{(1)}[P_s] = \perp$ and removes the now-nonviable path (P_s, P_i) from $M_i^{(1)}$.

From round 2 on, a party first forwards its accumulated information vector, then checks whether it can safely terminate. That is, whether it can already guarantee that every other honest party will eventually terminate on the same value. If $|V_i^{(r-1)}| > 1$, the sender must have signed two conflicting values, so P_i outputs $\perp$ (which can be mapped to any default value). Otherwise, P_i checks whether in round r any path of length at least $r+1$ remains in $M_i^{(r-1)}$. Since only paths of length at least $r+1$ could still be in transit. Once none remain, P_i knows no later round could add new information, and outputs whichever value it has observed in $V_i^{(r-1)}$, or `NoMessage` (which can be mapped to any default value) if it has not observed any.

If P_i has not terminated, it then receives a message from every other party P_k and updates $\mathsf{I}_i^{(r)}[P_k]$ accordingly, or records $\perp$ if P_k sent nothing valid. The party then uses observed accusations to prune paths from M_i exactly as described in Section 1.3. Finally, it can extend V_i with any newly-observed value of the sender over a viable path.

Protocol Π_{base}

Assume sender $P_s \in \mathcal{P}$ has input x.

Initialization $r = 0$, every party $P_i \in \mathcal{P} \setminus \{P_s\}$ does:

1: P_i sets $M_i^{(0)} = \{\pi | \pi$ being any path of distinct parties from P_s to P_i with length $\leq t+2\}$
2: P_i sets $V_i^{(0)} = \varnothing$

In round $r = 1$, sender P_s does the following:

1: Sender P_s computes $\sigma = \mathsf{Sgn}_{sk_s}(x)$, and sends $(0, x, \sigma)$ to all parties and terminates on output x.

In round $r = 1$, party $P_i \in \mathcal{P} \setminus \{P_s\}$ does the following:

Receive message $m = (0, x, \sigma)$ from the sender P_s:
  If $\texttt{checkValidity}(1, m)$, set $\mathsf{I}_i^{(1)}[P_s] = m$
  If no valid message is received, set $\mathsf{I}_i^{(1)}[P_s] = \perp$
If $\mathsf{I}_i^{(1)}[P_s] \neq \perp$:
  Set $M_i^{(1)} = M_i^{(0)}$ and $V_i^{(1)} = \{m.\mathsf{val}\}$
Else:
  Set $M_i^{(1)} = M_i^{(0)} \setminus \{(P_s, P_i)\}$ and $V_i^{(1)} = V_i^{(0)}$

In rounds $r = 2, \ldots, t+2$, party $P_i \in \mathcal{P} \setminus \{P_s\}$ does the following:

Send $m_i = (r-1, \mathsf{I}_i^{(r-1)}, \mathsf{Sgn}_{sk_i}(\mathsf{I}_i^{(r-1)}))$ to all parties
Check termination conditions (evaluated in order):
  (i) terminate on output $\perp$ if $|V_i^{(r-1)}| > 1$
  (ii) terminate on output x for $x \in V_i^{(r-1)}$ if no path of length $\geq r+1$ is left in $M_i^{(r-1)}$
  (iii) terminate on output $\texttt{NoMessage}$ if no path of length $\geq r+1$ is left in $M_i^{(r-1)}$

Receive message m_k from P_k:
  If $\texttt{checkValidity}(r, m_k)$, set $\mathsf{I}_i^{(r)}[P_k] = m_k$
  If no valid message is received, set $\mathsf{I}_i^{(r)}[P_k] = \perp$
Set: $M_{init-i}^{(r)} = M_i^{(r-1)}$ and $V_{init-i}^{(r)} = V_i^{(r-1)}$
For any path $\pi = (P_j, P_k, \ldots, P', P_i)$ such that $\mathsf{I}_i^{(r)}[\pi] = \perp$:
  Set $M_{init-i}^{(r)} = M_{init-i}^{(r)} \setminus \{\pi' | \pi' \text{ is a message path with suffix } \pi\}$
Set $M_i^{(r)} = M_{init-i}^{(r)}$
For any path $\pi = (P_s, \ldots, P_i) \in M_i^{(r)}$ with $len(\pi) = r+1$:
  Set $V_{init-i}^{(r)} = V_{init-i}^{(r)} \cup \{\mathsf{I}_i^{(r)}[\pi].\mathsf{val}\}$
Set $V_i^{(r)} = V_{init-i}^{(r)}$

Theorem 27. *Assuming a plain PKI, Protocol Π_{base} is a t-secure broadcast protocol that runs in $t+2$ rounds.*

We prove Theorem 27 in Section 5.3.

5.3 Proof of Correctness

We can now prove, that this protocol achieves broadcast (without achieving early termination). We call honest parties alive in some round r, if they do not terminate in round r (or any previous one).

Propagation. We prove a sequence of lemmas that culminate in termination, validity, and agreement. First, Lemma 29 shows that an honest party's messages always propagate reliably to every other honest party; Lemma 30 then shows that a path survives at every honest party as long as no evidence contradicts it. Combining the two, Lemma 31 shows that once one honest party observes a value, every other honest party either observes the same value within one round or has already terminated. Building on this, we then establish *value agreement*: Lemma 32 and Lemma 33 show that, once no candidate path remains (at the latest, by the final round), every honest party's set of observed values is fixed and identical – from which termination, validity, and agreement follow directly.

First we prove that an honest party will accept the initial message of an honest sender.

Lemma 28. *If the sender P_s is honest with input x, then every honest party P_i sets $\mathsf{I}_i^{(1)}[P_s] = (0, x, \sigma)$, i.e. accepts the sender's message.*

Proof. Since P_s is honest, it computes $\sigma = \mathsf{Sgn}_{sk_s}(x)$ and sends $(0, x, \sigma)$ to all parties in round 1. Every honest party P_i receives this message and calls $\texttt{checkValidity}(1, (0, x, \sigma))$, which returns $\mathsf{Vfy}_{pk_s}(x, \sigma) = \texttt{true}$ by construction. Hence P_i sets $\mathsf{I}_i^{(1)}[P_s] = (0, x, \sigma)$. □

The next lemma proves the analogous fact for any later round: an honest party's message will always be accepted by every other honest party (i.e. is considered to be valid).

Lemma 29. *Let P_i, P_j be honest parties. Let $m_i = (r-1, \mathsf{I}_i^{(r-1)}, \mathsf{Sgn}_{sk_i}(\mathsf{I}_i^{(r-1)}))$ be the message that P_i sends in round $r \geq 2$. Assuming P_j is alive in round r, then $\mathsf{I}_j^{(r)}[P_i] = m_i$, i.e. P_j deems the message to be valid and accepts it.*

Proof. Let $m_i = (r-1, \mathsf{I}_i^{(r-1)}, \mathsf{Sgn}_{sk_i}(\mathsf{I}_i^{(r-1)}))$ be the message that P_i sends in round $r \geq 2$, and assume P_j is alive in round r.

In round r, P_j receives m_i and checks its validity by calling $\texttt{checkValidity}(r, m_i)$. Since $r \geq 2$, this call falls into the `else` branch of Protocol `checkValidity`: the signature check succeeds because P_i is honest and signed $\mathsf{I}_i^{(r-1)}$ correctly.

If $r = 2$, the branch checks $\mathsf{I}_i^{(1)}[P_s]$: this is either $\perp$ (if P_i never received a valid message from the sender in round 1), in which case the check succeeds trivially, or the sender's own validly-signed message, which P_i only ever stores after having validated it itself. So the recursive call succeeds for the same reason it did when P_i first validated it.

If $r > 2$, the branch recurses on $\mathsf{I}_i^{(r-1)}[P_k]$ for every $P_k \in \mathcal{P} \setminus \{P_s\}$ with $\mathsf{I}_i^{(r-1)}[P_k] \neq \perp$. Since P_i is honest, it only ever stores an entry there after it successfully validates the message via `checkValidity` in round $r-1$. Since `checkValidity` is deterministic, the same recursive validation succeeds again for P_j.

Thus $\texttt{checkValidity}(r, m_i)$ returns `true`, and P_j sets $\mathsf{I}_j^{(r)}[P_i] = m_i$. □

The next lemma captures the idea that an honest party P_j cannot remove a message path $\pi' = (P_s, .., P_i, P_j)$ from M_j, as long as the prefix path $\pi = (P_s, .., P_i) \in M_i$. This means, if P_i still considers path π to potentially deliver the sender's message, then P_j also needs to consider the extension of it. The following lemmas then describe how the sender's message is propagated among the honest parties.

Lemma 30. *Let P_i, P_j be honest parties alive in round $r < t+1$. Assuming in round r party P_i contains the path $\pi = (P_s, .., P_i)$ in $M_i^{(r)}$, such that $len(\pi) = r+1$ and $P_j \notin \pi$. Then for any round $r' \leq r+1$ where P_j is alive, it holds that $(\pi, P_j) \in M_j^{(r')}$.*

Proof. If $\pi \in M_i^{(r)}$, it means that along that path π, there was no sign of corruption, i.e. there is no suffix π_1 of π such that $\mathsf{I}_i^{(r)}[\pi_1] = \perp$. Since P_i and P_j are honest, in round $r+1$, party P_j receives $m_i = (r, \mathsf{I}_i^{(r)}, \sigma_i)$ and sets $\mathsf{I}_j^{(r+1)}[P_i] = m_i$ by Lemma 29. Since there is no suffix π_1 of π such that $\mathsf{I}_i^{(r)}[\pi_1] = \perp$, there also cannot be a suffix π_2 of (π, P_j) such that $\mathsf{I}_j^{(r+1)}[\pi_2] = \perp$, as otherwise $\mathsf{I}_j^{(r+1)}[\pi_2] = \perp$ would imply that π_1 exists such that $\mathsf{I}_i^{(r)}[\pi_1] = \perp$.

Since $len(\pi) \leq t+1$, this implies that $len(\pi, P_j) \leq t+2$, and we know that the path (π, P_j) is in $M_j^{(0)}$ (as $P_j \notin \pi$). Since no suffix of (π, P_j) is $\perp$ by round $r+1$, the path cannot have been pruned, so as long as P_j is alive in round $r+1$, $(\pi, P_j) \in M_j^{(r+1)}$.

Also, since $M_j^{(r+1)} \subseteq M_j^{(r)}$ (paths are only removed during the execution, never added), it also holds that $(\pi, P_j) \in M_j^{(r')}$ for any $r' \leq r+1$ that P_j is alive. □

We can now prove that if an honest party P_i observes a value x signed by the sender for the first time in some round r, then P_j will also observe that value in the following round, or terminate in or before round $r+1$.

Lemma 31. *Let P_i, P_j be honest parties. Let $r < t+1$ be the first round such that $x \in V_i^{(r)}$. Then, either P_j has terminated on $\perp$ before or in round $r+1$, or for any round $r' \geq r+1$ where P_j is alive it holds that $x \in V_j^{(r')}$.*

Proof. Let $r < t+1$ be the first round such that $x \in V_i^{(r)}$, and let $\pi \in M_i^{(r)}$ be a path such that $len(\pi) = r+1$ and $\mathsf{I}_i^{(r)}[\pi].\mathsf{val} = x$. If P_j has terminated by round $\leq r+1$, it could only have terminated on $\perp$: by Lemma 30, $(\pi, P_j) \in M_j^{(r)}$ as long as P_j is alive through round r, and since this path has length $r+2 \geq (r+1)+1$, the path-exhaustion termination conditions (ii)/(iii) cannot trigger for P_j at or before round $r+1$.

Otherwise, by Lemma 30 we know that $(\pi, P_j) \in M_j^{(r+1)}$. Thus in round $r+1$, P_j receives the message $m_i = (r, \mathsf{I}_i^{(r)}, \sigma_i)$ and $\mathsf{I}_j^{(r+1)}[(\pi, P_j)].\mathsf{val} = x$, which implies $x \in V_j^{(r+1)}$.

Also, since $V_j^{(r')} \subseteq V_j^{(r'+1)}$ (values are only added during the execution, never removed), it also holds that $x \in V_j^{(r')}$ for any $r' \geq r+1$. □

Value Agreement. Next, we will prove that if parties have not terminated by the last round, then they hold the same value sets, which implies that they terminate on the same value in the last round.

Lemma 32. *Let P_i, P_j be honest parties that terminate in round $r = t+2$. Then, $V_i^{(r-1)} = V_j^{(r-1)}$.*

Proof. Let P_i, P_j be honest parties that terminate in round $r = t+2$. Let $x \in V_i^{(r-1)} = V_i^{(t+1)}$. We will prove that this implies that $x \in V_j^{(r-1)} = V_j^{(t+1)}$ by case distinction on the round that x was added to V_i.

Case 1: Let $r' < t+1$ be the first round such that $x \in V_i^{(r')}$. By Lemma 31 this implies that by round $r'+1 \leq t+1$ we have $x \in V_j^{(r'+1)}$ and thus $x \in V_j^{(t+1)}$.

Case 2: Let $r' = t+1$ be the first round such that $x \in V_i^{(r')}$ and let π be the path such that $\pi \in M_i^{(r')}, len(\pi) = r'+1 = t+2$ and $\mathsf{I}_i^{(r')}[\pi].\mathsf{val} = x$. Given the length of π, there must be at least one other honest party P_k in π. This implies that there is a round $r'' < t+1$ such that $x \in V_k^{(r'')}$. By Lemma 31 this implies that by round $r''+1 \leq t+1$ we have $x \in V_j^{(r''+1)}$ and thus $x \in V_j^{(t+1)}$. □

The previous lemma handled the special case of the very last round, $r = t+2$. We now generalize this to any round in which a party runs out of candidate paths, showing that value agreement occurs as soon as this happens and not only at the final round. This also means that honest parties can terminate multiple rounds apart, but their value-sets are fixed as soon as the first honest party terminates.

Lemma 33. *Let P_i, P_j be honest parties. Let $r < t+2$ be the first round s.t. there is no path $\pi \in M_i^{(r-1)}$ with $len(\pi) \geq r+1$. Then, assuming that P_j is alive in round r, $V_j^{(r-1)} \subseteq V_i^{(r-1)} = V_j^{(r')}$ for every $r' \geq r$.*

Proof. Let $r < t+2$ be the first round s.t. there is no path $\pi \in M_i^{(r-1)}$ s.t. $len(\pi) \geq r+1$. This would imply that party P_i terminates in round r.

Assume by contradiction that in some round $r' \geq r-1$, honest party P_j adds a new value x to $V_j^{(r')}$, where $x \notin V_i^{(r-1)}$. This means, there is a path π^* such that $\pi^* \in M_j^{(r')}$, $len(\pi^*) = r'+1$ and $\mathsf{I}_j^{(r')}[\pi^*].\mathsf{val} = x$. Furthermore, it is clear that $P_i \notin \pi^*$, as otherwise P_i would have seen the value x and added x to V_i.

If $r' < t+1$, we can use Lemma 30 to show that for any $r'' \leq r'+1$ it must hold that $(\pi^*, P_i) \in M_i^{(r'')}$ and $len(\pi^*, P_i) \geq r+1$. This leads to a contradiction with the assumption that there is no path $\pi \in M_i^{(r-1)}$ s.t. $len(\pi) \geq r+1$.

If $r' = t+1$, then there must be an honest party $P_k \in \pi^*$ (besides P_j). Using Lemma 30 with the shortened π^* (only up to the P_k), we would again get a contradiction to the assumption, that there is no path $\pi \in M_i^{(r-1)}$ s.t. $len(\pi) \geq r+1$.

This implies that $V_j^{(r')} \subseteq V_i^{(r-1)}$ for every $r' \geq r-1$. Thus, it remains to show that $V_i^{(r-1)} \subseteq V_j^{(r)}$.

Let $x \in V_i^{(r-1)}$, and let $r'' \leq r-1$ be the round at which x first appeared in V_i . Since $r < t+2$, we have $r'' \leq r-1 < t+1$, so Lemma 31 applies directly: $x \in V_j^{(r''+1)} \subseteq V_j^{(r)}$ (since $r''+1 \leq r$).

Thus, if some honest party P_i no longer considers any message paths in some round r, then from the following round on, the value-sets of all honest parties are fixed. □

Termination, Validity, Agreement. With propagation and value agreement established, we can now directly prove that Π_{base} satisfies the three properties required for broadcast: termination, validity, and agreement.

Lemma 34. *Protocol Π_{base} achieves termination.*

Proof. Parties add paths of length up to $t+2$ into their set M_i at initialization. Thus, there is never a path of length $t+3$ in their respective M_i. Therefore, latest in round $t+2$, either condition (ii) or (iii) will trigger, and the party will terminate, as there is no path of length $\geq t+3$ in M_i. □

Note that, we could modify the protocol to evaluate the termination-conditions at the end of round $t+1$ too, to achieve termination in $t+1$ rounds. The reason why this would not break any arguments follows from the fact, that any party would terminate latest in round $t+2$, and hence would not receive any of the messages sent that round. That implies, that this decision could be reached one round earlier.

We next show validity.

Lemma 35. *Protocol Π_{base} achieves validity.*

Proof. Assuming the sender is honest, it will send $(0, x, \sigma)$ to all parties in round 1, and by Lemma 28, every honest party sets $\mathsf{I}_i^{(1)}[P_s] = (0, x, \sigma)$ and adds x to $V_i^{(1)}$. Given that the sender is honest, it will not sign a different message, so no honest party ever adds anything other than x to its value set V_i. By Lemma 34, every honest party terminates; since V_i is never empty and never contains more than one element from round 1 on, termination always occurs via condition (ii), outputting x. □

Finally, we show agreement.

Lemma 36. *Protocol Π_{base} achieves agreement.*

Proof. Let P_i, P_j be two honest parties, and let P_i be the party that terminates first (or they terminate in the same round). We will prove agreement by case distinction on the output of P_i and when the parties terminate.

Case 0: Assume P_i terminates in round $r = t + 2$. Given that P_i terminates first, we know that the other honest party P_j also terminates in round $t + 2$. Applying Lemma 32 we know that $V_i^{(t+1)} = V_j^{(t+1)}$ and thus both parties will terminate on the same value.

Case 1: Assume P_i terminates in round $r < t + 2$ on $\perp$ because for the first time $|V_i^{(r-1)}| > 1$. If $|V_j^{(r-1)}| > 1$, then P_j also terminates on $\perp$ in the same round. Else, applying Lemma 31 on round $r - 1$ we can show that for any element $x \in V_i^{(r-1)}$ we have $x \in V_j^{(r)}$ and thus in round $r + 1$ $|V_j^{(r)}| > 1$, and P_j terminates on $\perp$ one round later. In either case, both parties terminate on $\perp$.

Case 2: Assume P_i terminates in round $r < t + 2$ because for the first time it does not contain a path of length $\geq r + 1$ in $M_i^{(r-1)}$. Then by Lemma 33 we know that $V_j^{(r-1)} \subseteq V_i^{(r-1)} = V_j^{(r')} \forall r' \geq r$. If P_j terminates the same round r, we can apply Lemma 33 once more to receive $V_i^{(r-1)} \subseteq V_j^{(r-1)}$, which implies that $V_j^{(r-1)} = V_i^{(r-1)}$ and they agree on their outputs. Note that P_j cannot terminate on $\perp$ in round r because that would contradict $V_j^{(r-1)} \subseteq V_i^{(r-1)}$.
If P_j terminates in round $> r$, then P_i and P_j will have equal outputs as $V_i^{(r-1)} = V_j^{(r')} \forall r' \geq r$. □

Theorem 27. *Assuming a plain PKI, Protocol* Π_{base} *is a* t*-secure broadcast protocol that runs in* $t + 2$ *rounds.*

Proof. Follows from Lemma 34, Lemma 35 and Lemma 36. Note that modifying the last round to evaluate the termination condition twice would lead to a protocol running in $t + 1$ rounds. □

6 Broadcast Protocol with Asymptotically Optimal Early Termination

Section 5.3 showed that Π_{base} achieves broadcast, but doesn't necessarily terminate early: a message path whose intermediate parties are all honest is never pruned from an honest party's path set M, since honest parties never accuse one another (Lemma 29). This is exactly the gap anticipated in Section 1.3: pruning alone is not sufficient for early termination, since executions with few or no faults trigger no accusations, and thus eliminate no paths. Even in the *best-case* execution, where every party (including the sender) is honest, Π_{base} is forced to run the full $t + 2$ rounds.

Concretely, set $n = 7, t = 5$ with sender $P_s = P_7$ and $P_1, \ldots, P_6$ all honest. Every party initializes M with all paths of length up to $t + 2 = 7$, so P_1's set contains, e.g., $\pi = (P_7, P_6, P_5, P_4, P_3, P_2, P_1)$. Since every hop along π is honest, no party ever accuses another, and π is never pruned. In that case, P_1 cannot terminate until round $t + 2 = 7$, exactly the worst-case bound. Scaling to $n = 100, t = 98$ with everyone honest, parties would need to run the protocol for all 100 rounds. Π_{base} has no mechanism for removing in some sense good paths – paths along which every hop looks legitimate.

To fix this without weakening the trust honest parties place in each other, we allow parties to prune additional paths whenever they observe consistent behavior along them. This process we call *path augmentation*. Recall that Π_{base} already allows parties to prune paths whenever an accusation is observed along them, and now to complement, we allow parties to prune paths if they observe a hop along them that is augmentable (Section 1.3).

The protocol we present here, Π_{fast}, is the first to achieve early-terminating broadcast for any $t < n$ (rather than only $t < (1 - \epsilon)n$, as in prior work), terminating in $O(f)$ rounds whenever

f parties are corrupted. Like Π_{base}, it relies on exponential information gathering and so incurs exponential communication and computation complexity; unlike Π_{base}, each round is now split into several phases rather than one, since parties must first announce what they intend to send before sending the actual message.

Concretely, each round of Π_{fast} follows this pattern:

1. Announce, via a form of weak broadcast, what will be sent this round.
2. Forward the accumulated information from the previous round.
3. Evaluate termination.
4. Receive incoming messages, and validate them: both for correct format and for consistency with what was announced.
5. Use the newly-validated messages to update M_i and V_i, pruning both bad paths (where a hop failed) and good paths (where a hop is augmentable).

The remainder of this section is organized as follows. Section 6.1 introduces the additional notation needed to announce messages and construct path augmentations; Section 6.2 presents Π_{fast} itself together with the round-by-round intuition behind it; and Section 6.3 proves that Π_{fast} achieves early termination, validity, and agreement.

6.1 Protocol-Specific Preliminaries

Simplifying Assumptions. As in Section 5.1, we assume without loss of generality that a corrupted party never sends a message of the wrong format or multiple messages to the same receiver within a single round. Any party that does so can be treated by every recipient as having sent nothing at all ($\perp$) that round. Furthermore, to simplify the presentation, we assume that every message is implicitly signed by its author, and each signature is checked by the recipients, and we omit signatures from the protocol descriptions below.

Messages, the Information Vector, and Announcement Vectors. Every message in Π_{fast} carries, in addition to the information vector I already introduced in Section 5.1, a record of what its author announced and observed via weak broadcast before sending, the grade that announcements received, and a set of parties that confirm each announcement. We need this because path augmentation requires a party to prove not just what it received, but that another party can independently verify having witnessed the exact same announcement, something I alone cannot certify.

Formally, every message now has the form $(r, \mathsf{I}, \mathsf{WB}, \mathsf{G}, \mathsf{Conf}, \sigma)$, where WB, G, and Conf are new vectors, indexed by parties exactly like I. So $\mathsf{WB}_i[P_j]$ is the message P_i obtained as the output of the weak broadcast in which P_j acted as sender, $\mathsf{G}_i[P_j] \in \{0, 1\}$ is the corresponding grade, and $\mathsf{Conf}_i[P_j]$ is the set of (party, signed message) pairs certifying who else confirmed the same announcement, all output of Appendix A's Confirming Weak Broadcast, run with P_j as sender.

Similarly to before, the information vector $\mathsf{I}_i^{(r)}$ of party P_i in round r is structured as follows:

- If $r = 1$, the vector $\mathsf{I}_i^{(1)}$ contains only 1 element, namely the message received from the sender: $\mathsf{I}_i^{(1)}[P_s] = m$, where m is either the message $(0, x)$ (the sender's input and implicitly also its signature) or $m = \perp$, denoting that no valid message was received. For all other parties the element is not defined which is distinct from being defined as $\perp$.

- If $r > 1$, $\mathsf{I}_i^{(r)}$ has one entry for every party $P_j \in \mathcal{P} \setminus \{P_s\}$: $\mathsf{I}_i^{(r)}[P_j]$ is the message P_i received from P_j in round r, or $\perp$ if none (or an invalid one) was received.

The new vectors $\mathsf{WB}_i^{(r)}$, $\mathsf{G}_i^{(r)}$, and $\mathsf{Conf}_i^{(r)}$ have $n-1$ elements from the second round on, where as mentioned previously, $\mathsf{WB}_i^{(r)}[P_j]$ will contain the message, $\mathsf{G}_i^{(r)}[P_j]$ will hold the grade, and $\mathsf{Conf}_i^{(r)}[P_j]$ will hold the confirmation set that P_i computed during weak broadcast with P_j as a sender for round r messages.

We extend our access-operator notation so that, given a message $\mathsf{I}_i^{(r)}[P_j]$ recorded from P_j, we can directly read off its I-, WB-, G-, or Conf-component:

Protocol $\mathsf{I}_i^{(r)}[P_j].T$

1: $\mathsf{I}_i^{(r)}[P_j] = m_j = (r-1, \mathsf{I}_j, \mathsf{WB}_j, \mathsf{G}_j, \mathsf{Conf}_j)$
2: If $T = \mathsf{I}$: return I_j
3: If $T = \mathsf{WB}$: return WB_j
4: If $T = \mathsf{G}$: return G_j
5: If $T = \mathsf{Conf}$: return Conf_j

Message paths and the path-access operator $\mathsf{I}_i[\pi]$ carry over unchanged in spirit from Section 5.1, now simply returning the extended tuple $(r, \mathsf{I}, \mathsf{WB}, \mathsf{G}, \mathsf{Conf})$ at each hop instead of just (r, I) and recursing into the information vector directly. The value-extraction operator $\mathsf{I}_i[\pi].\mathsf{val}$ also carries over the same way, extracting x from the sender's round-1 message $(0, x)$ (or $\perp$ from $\perp$). Protocol $\mathsf{I}[\pi]$ restates the recursive definition for completeness.

Protocol $\mathsf{I}_i[\pi]$

If $\pi = (P_j)$.
1: Return $\mathsf{I}_i[P_j]$
Else let $\pi = (P_1, P_2, \ldots, P_k, P_l)$.
2: If $P_l = P_i$:
3: Return $\mathsf{I}_i[(P_1, P_2, \ldots, P_k)]$
4: Else:
5: Let $\mathsf{I}_i[P_l] = m_l = (r, \mathsf{I}_l, \mathsf{WB}_l, \mathsf{G}_l, \mathsf{Conf}_l)$
6: Return $\mathsf{I}_l[(P_1, P_2, \ldots, P_k)]$

Confirming Weak Broadcast. Before sending its round-r message, every party runs $n-1$ parallel instances of *Confirming Weak Broadcast* (Appendix A): once as sender, announcing what it is about to send, and once as receiver in every other party's instance. Like Weak Broadcast [11], Confirming Weak Broadcast satisfies weak consistency: if two honest parties both assign grade 1 to a sender's announcement, their outputs for it agree. Beyond this, it also guarantees confirmation: whenever an honest party assigns grade 1, it additionally obtains a set C of signed proofs of other parties that confirmed receiving the identical announcement. It is this set of signed proofs that lets a party far down a relay chain verify, without ever having directly witnessed the original announcement itself, that the value it is being asked to relay cannot have been substituted along the way. Finally, a strengthened validity property guarantees that if the sender is honest, then each party receives its message, has grade 1, and has every honest party in its confirming set.

Validity and Consistency of Messages. Checking that a received message is valid now additionally means checking that it matches what its author announced: a corrupted party can no longer send different content to different honest parties while claiming it is the same message,

since any divergence would be caught against the weak-broadcast record. Protocol `checkValidity` makes this precise.

Concretely, Protocol `checkValidity` performs four groups of checks on a received message $m_k = (r-1, \mathsf{I}_k^{(r-1)}, \mathsf{WB}_k^{(r-1)}, \mathsf{G}_k^{(r-1)}, \mathsf{Conf}_k^{(r-1)})$. Lines 2–3 confirm that P_k's own announcement, as P_i witnessed it via weak broadcast with P_k as sender, matches what P_k actually sent and was confirmed (grade 1). Line 4 handles the base case $r = 2$, where the only thing left to verify is what P_k itself claims to have received from the sender in round 1. Lines 6–9 cross-check P_k's claims about every third party P_j: Line 6 verifies that if both P_k and P_i claim grade-1 confidence in P_j's value, they agree, as guaranteed by the weak consistency property of Definition 54. Lines 7–9 mirror the confirmation property of Definition 54 directly against P_k's claimed $\mathsf{Conf}_k^{(r-1)}[P_j]$: grade 0 must come with an empty confirmation set, grade 1 must include the sender P_j itself, and every party named in the set must carry a signature that actually verifies on the claimed value. Lines 10–12 finally recurse into the rest of P_k's claimed history, checking that every message it built upon was itself valid and consistent.

Note that a message is only ever checked for validity after the announcement is received, so we may always assume we are checking a full tuple $(r, \mathsf{I}, \mathsf{WB}, \mathsf{G})$; the messages exchanged within a weak broadcast do not themselves need validating, since they never directly update a party's M or V.

Protocol `checkValidity`(P_i, r, m_k)

Party P_i checks the message m_k received from P_k in round r:

If $r = 1$, return `true` if the message is correctly signed by the sender $P_k = P_s$. Else return `false`.
If $r > 1$, let $m_k = (r-1, \mathsf{I}_k^{(r-1)}, \mathsf{WB}_k^{(r-1)}, \mathsf{G}_k^{(r-1)}, \mathsf{Conf}_k^{(r-1)})$ with a valid signature by P_k:

1: Check the following
2: If not $\mathsf{WB}_i^{(r-1)}[P_k] == \mathsf{I}_k^{(r-1)}$: return `false`
3: If not $\mathsf{G}_i^{(r-1)}[P_k] == 1$: return `false`
4: If $r = 2$: return `true` if $\mathsf{I}_k^{(r-1)}[P_s] = \perp$, else return `checkValidity`$(P_k, r-1, \mathsf{I}_k^{(r-1)}[P_s])$
5: For each party $P_j \in \mathcal{P} \setminus \{P_s\}$:
6: If $\mathsf{G}_k^{(r-1)}[P_j] == 1$ and $\mathsf{G}_i^{(r-1)}[P_j] == 1$ and $\mathsf{WB}_k^{(r-1)}[P_j] \neq \mathsf{WB}_i^{(r-1)}[P_j]$: return `false`
7: If $\mathsf{G}_k^{(r-1)}[P_j] == 0$ and $\mathsf{Conf}_k^{(r-1)}[P_j] \neq \perp$: return `false`
8: If $\mathsf{G}_k^{(r-1)}[P_j] == 1$ and $P_j \notin \mathsf{Conf}_k^{(r-1)}[P_j]$: return `false`
9: If $\mathsf{G}_k^{(r-1)}[P_j] == 1$: for every $P_x \in \mathsf{Conf}_k^{(r-1)}[P_j]$, check that P_x's signature verifies on $\mathsf{WB}_k^{(r-1)}[P_j]$
10: For each party $P_j \in \mathcal{P} \setminus \{P_s\}$ s.t. $\mathsf{I}_k^{(r-1)}[P_j] \neq \perp$:
11: If not `checkValidity`$(P_k, r-1, \mathsf{I}_k^{(r-1)}[P_j])$: return `false`
12: If not `checkConsistencyParty`$(\mathsf{I}_k^{(r-1)}, P_j)$: return `false`
Return `true`

Beyond validity, a party must also check that the sender of a message it accepts has been consistent across all observed messages from the point of view of the receiver: an honest party will never sign two conflicting messages of the same type in the same round, nor fail to acknowledge a message it actually received, nor can it be named (with its own signature) in another party's Conf set for some value while signing a different value for that same announcement. Protocol `checkConsistencyParty` checks exactly these things.

Protocol `checkConsistencyParty`(I_i, P_k)

Party P_i checks that party P_k acts consistently:

1: In all of I_i, check for any two messages m_1, m_2 signed by P_k, of the same type (I, WB, G, or Conf)

and from the same round, that $m_1 = m_2$.

2: Check for all vectors $\mathsf{I}, \mathsf{WB}, \mathsf{G}, \mathsf{Conf}$ signed by P_k (in any round), that $\mathsf{I}[P_i] \neq \perp$, $\mathsf{WB}[P_i] \neq \perp$, $\mathsf{G}[P_i] = 1$, $\{P_i, P_k\} \subseteq \mathsf{Conf}[P_i]$.

3: Check for every vector Conf_x signed by any party P_x (in any round observed sofar by P_i in I_i), and for any party P_j such that $P_k \in \mathsf{Conf}_x[P_j]$, that $\mathsf{WB}_k[P_j] = \mathsf{WB}_x[P_j]$, i.e. if P_x has a signature in Conf_x confirming P_k, then P_k cannot have set a different value.

If all checks work out, return `true`. Else return `false`.

Path Augmentation. Recall from Section 1.3 the central idea we still need to formalize: if P_i has the guarantee that another party P_j observed the same announcement from some P_k as P_i did, then a path routed through P_k to P_j to P_i can be treated as equivalent to one routed through P_k to P_i directly. More generally, a path can be *augmented* whenever a whole cluster of parties all witness the same message on a given hop, and, can *mutually prove* this to one another via Conf, so that no party in the cluster can later be caught relaying a different value without contradicting its own earlier signature. Thus, a cluster is a set C of parties such that $C \subseteq \mathsf{Conf}_i$ for every party $P_i \in C$. Protocol `pathAugmentation` computes, for a given path π ending at P_i, the set of all paths that P_i's current information vector lets it treat as equivalent to π.

Protocol `pathAugmentation`(I_i, π)

Let $\pi = (P_1, P_2, \ldots, P_l, P_i)$.

1: Initialize sets $S_j = \{\varepsilon\}\ \ \forall j \in [l]$ [a]

2: For every $P_j, P_{j+1} \in \pi$ (two consecutive parties on π) and for any subset $C_j \subseteq \mathcal{P}$, such that $P_j, P_{j+1} \in C_j$:

3: Let r be the round in which P_{j+1} receives the message $\mathsf{I}_j^{(r-1)}$ from P_j along the path π: $\mathsf{I}_{j+1}^{(r)}[P_j].\mathsf{I} = \mathsf{I}_j^{(r-1)}$

4: Let $\mathsf{WB}_{j+1}^{(r-1)}, \mathsf{G}_{j+1}^{(r-1)}, \mathsf{Conf}_{j+1}^{(r-1)}$ be the vectors that P_{j+1} computes through the weak broadcasts at the beginning of round r

5: If $\forall P \in C_j$, where $\mathsf{WB}_P := \mathsf{I}_{j+1}^{(r)}[P].\mathsf{WB}$, $\mathsf{G}_P := \mathsf{I}_{j+1}^{(r)}[P].\mathsf{G}$, and $\mathsf{Conf}_P := \mathsf{I}_{j+1}^{(r)}[P].\mathsf{Conf}$, it holds $\mathsf{WB}_P[P_j] = \mathsf{I}_j^{(r-1)}$, $\mathsf{G}_P[P_j] = 1$, and $C_j \subseteq \mathsf{Conf}_P[P_j]$

6: then $S_j = S_j \cup \{\pi' \mid \pi' \text{ is any path through } C_j \setminus \{P_j, P_{j+1}\}\}$

7: Set $M = \{\pi' \mid \pi' = (P_1, \pi_1, P_2, \ldots, P_l, \pi_l, P_i)$ for any $\pi_j \in S_j, j \in [l]\}$

8: Return M

[a] By ε we denote the empty path

The new condition $C_j \subseteq \mathsf{Conf}_P[P_j]$ (checked for every $P \in C_j$) is what makes the augmentation *mutually* verifiable: every party in the cluster must be able to produce, from its own $\mathsf{Conf}_P[P_j]$, a signed proof that every other member of C_j confirmed the identical announcement message. Since the inclusion of a party is accompanied with its signature on that message, no member of C_j can later relay a different value without directly contradicting proof that is already in circulation.

Note that this can produce paths with repeating parties; these are harmless, since only paths in which every party is distinct are ever actually removed from an M_i set.

Message Paths and Observed Values. As in Section 5.1, each party P_i maintains a set M_i of plausible message paths and a set V_i of values it has observed signed by the sender. The bookkeeping around M_i is now a little richer, since a path can be removed for two different reasons within the same round: $M_{init-i}^{(r)}$ denotes the set at the start of round r, $M_{acc-i}^{(r)}$ the set once paths carrying an observed accusation (or an augmentation thereof) have been removed, and $M_i^{(r)}$ the final set at the end of the round, once paths augmenting an already-accounted-for value have also been removed.

6.2 The Protocol

We can now present Π_{fast} itself. Its overall shape is identical to Π_{base}'s: initialize M_i/V_i, let the sender speak once in round 1, then run a general round loop. The two differences are exactly the ones motivated above: every subsequent round now opens with a weak-broadcast announcement phase, and the pruning step at the end of a round removes paths for two distinct reasons (an observed accusation, or a redundant augmentation) rather than just for observed accusations.

In round 1, the sender sends its input directly, and every other party behaves exactly as in Π_{base}: it either sets $\mathsf{I}_i^{(1)}[P_s]$ to the sender's message and keeps $M_i^{(1)}$ intact, or marks P_s as $\perp$ and discards the now nonviable path (P_s, P_i).

From round 2 on, a party first participates in $n-1$ parallel weak broadcasts (once as sender, announcing its own accumulated information; once as receiver for every other party), then forwards its full message $(r-1, \mathsf{I}_i^{(r-1)}, \mathsf{WB}_i^{(r-1)}, \mathsf{G}_i^{(r-1)}, \mathsf{Conf}_i^{(r-1)})$, and only then checks whether it may terminate. The checks for termination proceed exactly as in Π_{base}, using $V_i^{(r-1)}$ and $M_i^{(r-1)}$ to determine if it may terminate. If the party has not terminated, it receives and validates the incoming messages (checking both correctness and consistency), and updates M_i in two stages: first, $M_{acc-i}^{(r)}$ removes every path along which either an accusation, or an augmentation of an accusation, was observed; then $M_i^{(r)}$ further removes, for every value newly added to $V_i^{(r)}$, every path that is an augmentation of the path that value was observed on.

Protocol Π_{fast}

Assume sender $P_s \in \mathcal{P}$ has input x.

Initialization $r = 0$, every party $P_i \in \mathcal{P} \setminus \{P_s\}$ does:

1: P_i sets $M_i^{(0)} = \{\pi | \pi$ being any path of distinct parties from P_s to P_i with length $\leq t+2\}$
2: P_i sets $V_i^{(0)} = \varnothing$

In round $r = 1$, sender P_s does the following:

1: Sender P_s sends $(0, x)$ to all parties and terminates on output x.

In round $r = 1$, party $P_i \in \mathcal{P} \setminus \{P_s\}$ does the following:

1: Receive message $m = (0, x)$ from the sender P_s:
2: If $\texttt{checkValidity}(P_i, 1, m)$, set $\mathsf{I}_i^{(1)}[P_s] = m$
3: If no valid message is received, set $\mathsf{I}_i^{(1)}[P_s] = \perp$
4: If $\mathsf{I}_i^{(1)}[P_s] \neq \perp$:
5: Set $M_i^{(1)} = M_i^{(0)}$ and $V_i^{(1)} = \{m.\mathsf{val}\}$
6: Else:
7: Set $M_i^{(1)} = M_i^{(0)} \setminus \{(P_s, P_i)\}$ and $V_i^{(1)} = V_i^{(0)}$

In rounds $r = 2, \ldots, t+2$, party $P_i \in \mathcal{P} \setminus \{P_s\}$ does the following:

1: Participate in $n-1$ parallel instances of Confirming Weak Broadcast $\Pi_1^{\mathsf{wb}}, \ldots, \Pi_{n-1}^{\mathsf{wb}}$, where P_i acts as sender in instance Π_i^{wb} with input $\mathsf{I}_i^{(r-1)}$, and as receiver in all other instances.
2: Let $(\mathsf{WB}_i^{(r-1)}[P_j], \mathsf{G}_i^{(r-1)}[P_j], \mathsf{Conf}_i^{(r-1)}[P_j])$ be the output of instance Π_j^{wb}.
3: Send $m_i = (r-1, \mathsf{I}_i^{(r-1)}, \mathsf{WB}_i^{(r-1)}, \mathsf{G}_i^{(r-1)}, \mathsf{Conf}_i^{(r-1)})$ to all parties.
4: Check termination conditions (evaluated in order):
5: (i) terminate on output $\perp$ if $|V_i^{(r-1)}| > 1$;
6: (ii) terminate on output x, for $x \in V_i^{(r-1)}$, if no path of length $\geq r+1$ remains in $M_i^{(r-1)}$;
7: (iii) terminate on output $\texttt{NoMessage}$ if no path of length $\geq r+1$ remains in $M_i^{(r-1)}$.

Receive message m_k from every party P_k:
    Set $\mathsf{I}_i^{(r)}[P_k] = m_k$, or $\perp$ if none was received.
For every P_k with $\mathsf{I}_i^{(r)}[P_k] \neq \perp$:
    If not $\texttt{checkValidity}(P_i, r, \mathsf{I}_i^{(r)}[P_k])$, set $\mathsf{I}_i^{(r)}[P_k] = \perp$
For every P_k with $\mathsf{I}_i^{(r)}[P_k] \neq \perp$:
    If not $\texttt{checkConsistencyParty}(\mathsf{I}_i^{(r)}, P_k)$, set $\mathsf{I}_i^{(r)}[P_k] = \perp$
Initialize: $M_{init-i}^{(r)} = M_i^{(r-1)}$ and $V_{init-i}^{(r)} = V_i^{(r-1)}$
Set $M_{temp} = M_{init-i}^{(r)}$
For any path $\pi = (P_j, P_k, \ldots, P', P_i)$ such that $\mathsf{I}_i^{(r)}[\pi] = \perp$:
    remove from M_{temp} all paths with suffix π or
    with suffix (P_j, π') for $\pi' \in \texttt{pathAugmentation}(\mathsf{I}_i^{(r)}, (P_k, \ldots, P', P_i))$
Set $M_{acc-i}^{(r)} = M_{temp}$
Set $V_{temp} = V_{init-i}^{(r)}$
For any path $\pi = (P_s, \ldots, P_i) \in M_{temp}$ with $len(\pi) = r + 1$:
    set $V_{temp} = V_{temp} \cup \{\mathsf{I}_i^{(r)}[\pi].\mathsf{val}\}$
    remove from M_{temp} all paths $\pi' \in \texttt{pathAugmentation}(\mathsf{I}_i^{(r)}, \pi)$
Set $M_i^{(r)} = M_{temp}$
Set $V_i^{(r)} = V_{temp}$

Intuitively, the first pruning pass is exactly Π_{base}'s: a suffix with an observed $\perp$ can never carry the sender's value, and neither can any path that is an augmentation of it (since augmentation only ever asserts that two paths carry the *same* information). The second pass is new: once a path has actually delivered a value into V_i, every path augmentation will carry the same value, and so can be discarded too.

Theorem 37. *Assuming a plain PKI, Π_{fast} is a t-secure broadcast protocol that runs in $O(f)$ rounds.*

We prove Theorem 37 in Section 6.3.

6.3 Proof of Correctness

We can now prove that this protocol achieves broadcast with early termination. We call honest parties alive in some round r, if they have not terminated before or in round r.

We first establish that messages propagate reliably, exactly as in Section 5.3 (Lemma 38), and that the new announcement mechanism prevents a corrupted party from telling two honest parties different things without being caught (Lemma 39), which is exactly what makes path augmentation sound at a single hop (Lemma 40, Lemma 41, Lemma 42). Combining these, we prove the central guarantee that makes pruning via path augmentation safe in the first place: an honest party sitting anywhere along an augmented chain can never disagree with the value the chain's owner originally observed, however many corrupted parties relay it in between (Lemma 43, Lemma 44). Building on this, we show that paths and values still propagate the way they did in Π_{base} (Lemma 45, Lemma 46), and that value sets stabilize across honest parties in the same way (Lemma 47, Lemma 48). We then prove a purely combinatorial fact about path augmentation, that any sufficiently long path is an augmentation of a short one, which is what lets us bound the round complexity by the actual number of corruptions f, rather than the worst case t. Combining all of this gives early termination, validity, and agreement.

Propagation. We first show that an honest party's message is always accepted by every other honest party: including the sender's very first message.

Lemma 38. *Let P_i, P_j be honest parties. Let m_i be the message that P_i sends in round r. Assuming P_j is alive in round r, then at the end of round r, $\mathsf{I}_j^{(r)}[P_i] = m_i$, i.e. P_j deems the message to be valid and accepts it.*

Proof. We prove this by induction on r.
Round 1: If the sender $P_i = P_s$ is honest, then it will send a correctly signed message $m_i = (0, x)$ to all parties, including P_j. Since the message is signed correctly, P_j will set $\mathsf{I}_j^{(1)}[P_i] = m_i$.
Round $r > 1$: Assume the statement holds for all honest, alive parties and all rounds $r' < r$; we show it holds for round r.

Let $m_i = (r-1, \mathsf{I}_i^{(r-1)}, \mathsf{WB}_i^{(r-1)}, \mathsf{G}_i^{(r-1)}, \mathsf{Conf}_i^{(r-1)})$ be the message that P_i sends to all parties in round r. Then P_j sets $\mathsf{I}_j^{(r)}[P_i] = m_i$ in round r (lines 8,9), and checks its validity: `checkValidity`(P_j, r, m_i) (line 11). The signature-and-format check preceding Protocol `checkValidity`'s numbered steps succeeds, as P_i is honest and signs m_i correctly. Checks on lines 2 and 3 succeed by validity of weak broadcast: since P_i is honest, it announces $\mathsf{I}_i^{(r-1)}$ truthfully, so P_j's own weak-broadcast output for P_i matches it and carries grade 1.

If $r = 2$, line 4 checks $\mathsf{I}_i^{(1)}[P_s]$: this is either $\perp$ (in which case the check succeeds immediately), or the sender's own validly-signed round-1 message, which P_i, being honest, only ever stores after having validated it itself. Since `checkValidity` is deterministic, the same check succeeds again.

If $r > 2$, line 6 cross-checks P_i's claims about any third party P_k against P_j's own view: it only fires when both P_i and P_j claim grade-1 confidence in P_k's value and disagree. Since P_i and P_j are both honest, the weak consistency property of Definition 54 guarantees that if both hold a grade-1 confirmation for the same announcement, their values agree, so the check can never fail for honest P_i, P_j, regardless of P_k's behavior.

Lines 7–9 check that P_i's claimed $\mathsf{Conf}_i^{(r-1)}[P_k]$ is well-formed, for every third party P_k. Since P_i is honest, this set was computed by honestly running Confirming Weak Broadcast as a receiver with P_k as sender, so the Confirmation property of Definition 54 applies directly: if P_i graded P_k's announcement 0, its confirmation set is empty (line 7); otherwise it contains P_k's own signature (line 8), and every party named in it carries a signature that genuinely verifies on the confirmed value (line 9). All three checks therefore succeed by construction.

Lines 10–12 then recurse, for every P_k with $\mathsf{I}_i^{(r-1)}[P_k] \neq \perp$, into `checkValidity`$(P_i, r-1, \mathsf{I}_i^{(r-1)}[P_k])$ and `checkConsistencyParty`$(\mathsf{I}_i^{(r-1)}, P_k)$. Both succeed for the same reason: P_i, being honest, only ever stores an entry there after successfully running these exact checks itself, and both checks are deterministic.

Either way, `checkValidity`(P_j, r, m_i) returns `true`.

It remains to check that P_i also passes `checkConsistencyParty`$(\mathsf{I}_j^{(r)}, P_i)$ (line 12), i.e. that, from P_j's point of view, P_i has behaved consistently. We go through Protocol `checkConsistencyParty`'s three checks in turn:

Line 1 (no two conflicting signed messages of the same type and round) holds since P_i is honest and never signs inconsistently.

Line 2 requires that every vector P_i has signed properly acknowledges P_j: $\mathsf{WB}_i[P_j] \neq \perp$ and $\mathsf{G}_i[P_j] = 1$ hold by validity of weak broadcast, since P_i and P_j are both honest; $\{P_j, P_i\} \subseteq \mathsf{Conf}_i[P_j]$ holds by the confirmation property, since honest, grade-1 P_i is confirmed together with P_j; and $\mathsf{I}_i[P_j] \neq \perp$ is exactly the induction hypothesis, applied to the pair (P_j, P_i) in place of (P_i, P_j).

Line 3 requires that whenever P_i is named, with its own signature, in some other party P_x's $\mathsf{Conf}_x[P_j]$ set, P_i's own signed $\mathsf{WB}_i[P_j]$ matches P_x's, which holds since P_i, being honest, only ever signs consistently with what it truly confirmed.

Thus `checkConsistencyParty`$(\mathsf{I}_j^{(r)}, P_i)$ returns `true`.

Thus, at the end of round r we have $\mathsf{I}_j^{(r)}[P_i] = m_i$. □

The next lemma captures the fact that two honest parties will not accept different messages in the same round from the same party. This serves to show intuitively, why the announcement messages are useful: It prevents the corrupted parties from sending inconsistent messages to two honest parties without being accused.

Lemma 39. *Let P_i, P_j be honest and alive parties in round $r > 1$. Then for all $P_k \in \mathcal{P}$, if $\mathsf{I}_i^{(r)}[P_k] \neq \perp$ and $\mathsf{I}_j^{(r)}[P_k] \neq \perp$ then $\mathsf{I}_i^{(r)}[P_k].\mathsf{I} = \mathsf{I}_j^{(r)}[P_k].\mathsf{I}$.*

Proof. Assume by contradiction that $\mathsf{I}_i^{(r)}[P_k].\mathsf{I} = m_1$ and $\mathsf{I}_j^{(r)}[P_k].\mathsf{I} = m_2$ and $m_1 \neq m_2$.
By `checkValidity` (lines 2, 3) this implies that $\mathsf{WB}_i^{(r-1)}[P_k] = m_1$ and $\mathsf{G}_i^{(r-1)}[P_k] = 1$ (resp. $\mathsf{WB}_j^{(r-1)}[P_k] = m_2$ and $\mathsf{G}_j^{(r-1)}[P_k] = 1$). But P_i and P_j are both honest parties who assign grade 1 to P_k's round-r weak-broadcast announcement, yet record different outputs $m_1 \neq m_2$ for it, directly contradicting weak consistency of weak broadcast (with P_k as sender). Hence it must hold that $m_1 = m_2$. □

In the next lemma, we capture under what conditions an honest party P_i will add another party P_j on the last segment of a path augmentation. Intuitively, P_i will only add P_j to the path augmentation if P_j has observed the same message from P_k.

Lemma 40. *Let P_i, P_j be honest and alive parties, and let $\pi = (\ldots, P_k, P_i)$ be any path with $P_j \notin \pi$. If $(\ldots, P_k, P_j, P_i) \in$ `pathAugmentation`$(\mathsf{I}_i^{(r)}, (\ldots, P_k, P_i))$, then $\mathsf{I}_j^{(r)}[P_k] = \perp$ or $\mathsf{I}_j^{(r)}[P_k].\mathsf{I} = \mathsf{I}_i^{(r)}[P_k].\mathsf{I}$.*

Proof. We can look at the contents of the message $m_j = (r-1, \mathsf{I}_j^{(r-1)}, \mathsf{WB}_j^{(r-1)}, \mathsf{G}_j^{(r-1)}, \mathsf{Conf}_j^{(r-1)})$ received by P_i, i.e. $m_j := \mathsf{I}_i^{(r)}[P_j]$. Given that $(\ldots, P_k, P_j, P_i)$ is a path augmentation of $(\ldots, P_k, P_i)$ (from the point of view of $\mathsf{I}_i^{(r)}$), we have the following guarantees: $\mathsf{WB}_j^{(r-1)}[P_k] = \mathsf{I}_i^{(r)}[P_k].\mathsf{I}$ and $\mathsf{G}_j^{(r-1)}[P_k] = 1$.

If $\mathsf{I}_j^{(r)}[P_k] = \perp$, we are done. Otherwise P_j accepts a message from P_k in round r, so the validity check has to hold true, i.e. $\mathsf{WB}_j^{(r-1)}[P_k] = \mathsf{I}_j^{(r)}[P_k].\mathsf{I}$ (line 2 of `checkValidity`). Combined with the guarantee from path augmentation above, both $\mathsf{I}_i^{(r)}[P_k].\mathsf{I}$ and $\mathsf{I}_j^{(r)}[P_k].\mathsf{I}$ equal $\mathsf{WB}_j^{(r-1)}[P_k]$, and hence are equal to each other: $\mathsf{I}_j^{(r)}[P_k].\mathsf{I} = \mathsf{I}_i^{(r)}[P_k].\mathsf{I}$.

□

The following lemma shows that this still holds even if P_j is inserted alongside other parties at the last hop, rather than being the only party added there.

Lemma 41. *Let P_i, P_j be honest and alive players. Let $\pi = (\ldots, P_k, P_i)$ be any path s.t. $P_j \notin \pi$. If there is a path $\pi' \in$ `pathAugmentation`$(\mathsf{I}_i^{(r)}, \pi)$, such that P_j is between P_k and P_i on the path, then $\mathsf{I}_j^{(r)}[P_k] = \perp$ or $\mathsf{I}_j^{(r)}[P_k].\mathsf{I} = \mathsf{I}_i^{(r)}[P_k].\mathsf{I}$.*

Proof. For P_j to appear in π' between P_k and P_i, there must be a set C (of all parties between P_k and P_i) satisfying the checks on line 5 of `pathAugmentation`, evaluated for the final hop (P_k, P_i) of π.

Since the check holds for every party $P \in C$, it also holds for the subset $\{P_k, P_j, P_i\} \subseteq C$, so $(\ldots, P_k, P_j, P_i) \in$ `pathAugmentation`$(\mathsf{I}_i^{(r)}, \pi)$. The claim now follows directly from Lemma 40. □

Lastly, as a small technical lemma, we want to show that if two honest parties agree on the message they received from a common third party P_k, they also agree on P_k's entire recorded history further back along any path.

Lemma 42. *Let P_i, P_j be honest and alive parties, and let $\pi = (\ldots, P_m, P_k)$ be any path with a nonempty prefix before P_k, such that $P_i, P_j \notin \pi$. If $\mathsf{I}_i^{(r)}[P_k].\mathsf{I} = \mathsf{I}_j^{(r)}[P_k].\mathsf{I}$, then $\mathsf{I}_i^{(r)}[\pi] = \mathsf{I}_j^{(r)}[\pi]$.*

Proof. Write $\mathsf{I}_k^{(r-1)} := \mathsf{I}_i^{(r)}[P_k].\mathsf{I} = \mathsf{I}_j^{(r)}[P_k].\mathsf{I}$ for this shared vector. By Protocol $\mathsf{I}[\pi]$, since π's last element is $P_k \neq P_i, P_j$ and has a nonempty prefix before it, both $\mathsf{I}_i^{(r)}[\pi]$ and $\mathsf{I}_j^{(r)}[\pi]$ look up their own entry for P_k, which agree on their .I-component by assumption, and then recurse into the shared vector $\mathsf{I}_k^{(r-1)}$ over the same remaining prefix $(\ldots, P_m)$. Since the path-access operator is a deterministic function of the vector it recurses into and the path, and both computations now share the identical vector and remaining prefix, they yield the same result: $\mathsf{I}_i^{(r)}[\pi] = \mathsf{I}_j^{(r)}[\pi]$. □

The lemmas above establish agreement at a single hop of a path augmentation, and lift that agreement one layer further back along a path. We now put these together to state the guarantee path augmentation actually needs: whichever honest party ends up somewhere along an augmented chain rooted at some party P_k, it can never disagree with the value the chain's owner originally observed from P_k – regardless of how many, possibly corrupted, parties relay it in between. We first prove this for the base case, where the augmented chain replaces the bare hop (P_k, P_j) directly; the proof proceeds by induction on the chain's length, with Lemma 41 and Lemma 42 providing the base case.

Lemma 43. *Let P_i, P_j be honest and alive parties and let P_k be a third party. Assume that in round $r \leq t+1$, P_j observes the message m_k from P_k to be valid (and correctly announced), setting $\mathsf{I}_j^{(r)}[P_k].\mathsf{I} = m_k$. Then for any path $\pi = (P_k, \ldots, P_i)$ with $len(\pi) = c \geq 2$ such that (π, P_j) is a path augmentation of (P_k, P_j), we have $\mathsf{I}_i^{(r+c-2)}[\pi].\mathsf{I} = m_k$ or $\mathsf{I}_i^{(r+c-2)}[\pi'] = \perp$ for some suffix π' of π.*

Proof. Write $x := \mathsf{I}_j^{(r)}[P_k].\mathsf{I} = m_k$. Since π has length c and consists of P_k, P_i, and $c-2$ further parties, we may enumerate these w.l.o.g. as $P_{c-2}, \ldots, P_2, P_1$ in the order they appear on π (so $\pi = (P_k, P_{c-2}, \ldots, P_1, P_i)$, with P_{c-2} adjacent to P_k and P_1 adjacent to P_i); this is purely notational and does not presuppose anything about who these parties are.

Base case $c = 2$ (so $\pi = (P_k, P_i)$, with no intermediate parties): by Lemma 40 this immediately gives $\mathsf{I}_i^{(r)}[P_k].\mathsf{I} = x$ or $\mathsf{I}_i^{(r)}[P_k] = \perp$.

Structural fact. For $c > 2$, since (π, P_j) is a path augmentation of the pair (P_k, P_j), unwinding Protocol `pathAugmentation` for this input gives a single cluster C of all parties on π and P_j. And every $P \in C$ satisfies $\mathsf{WB}_P^{(r-1)}[P_k] = x$, $\mathsf{G}_P^{(r-1)}[P_k] = 1$, and $C \subseteq \mathsf{Conf}_P^{(r-1)}[P_k]$. This tells each $P \in C$ that it has itself, honestly or not, signed x with grade 1 towards P_j, and furthermore every honest party in the set has observed everyone in C sign x.

We argue in a sliding window of three consecutive parties on π: once a party is confirmed to relay $(x, 1, \pi \subseteq \mathsf{Conf})$ in round r, this pins down the *value* (but not yet the grade) the next two parties must relay, and one round later pins down the *grade* of the next party too – at which point the window has advanced by one, and the argument repeats.

Anchor. In round $r+1$, P_i receives directly from P_j the message $m_j = (r, \mathsf{I}_j^{(r)}, \mathsf{WB}_j^{(r)}, \mathsf{G}_j^{(r)}, \mathsf{Conf}_j^{(r)})$. This message contains (in $\mathsf{I}_j^{(r)}$) the structural fact mentioned above, that every $P \in C$ satisfies $\mathsf{WB}_P^{(r-1)}[P_k] = x$, $\mathsf{G}_P^{(r-1)}[P_k] = 1$, and $C \subseteq \mathsf{Conf}_P^{(r-1)}[P_k]$. This also means, that for P_i to accept the message from P_1 in round $r+1$, it needs to show that P_1 signed x with grade 1 and an accompanying

set $\mathsf{Conf}_1 \supset C$ (in the appropriate position of the message). If this was not the case, P_i would reject the message and we would have $\perp$ for some suffix $\pi' = (P_1, P_i)$ of π.

So we assume that P_1 relays a message with the information $(x, 1, \pi \subseteq \mathsf{Conf}_1)$ contained.

General step. Suppose we have shown P_q relays $(x, 1, \pi \subseteq \mathsf{Conf}_s)$ (the case $q = 1$ is the anchor above). Since $\pi \subseteq \mathsf{Conf}_s$, and in particular $P_{q+1} \in \mathsf{Conf}_s$, P_q already holds P_{q+1}'s own signature on x from round r: this is exactly what `checkValidity`'s Conf-signature check (line 9) verified. So if P_{q+1} later relays a different value, P_q would hold conflicting P_{q+1}-signed claims, which would be caught by `checkConsistencyParty` and forcing P_q to reject, which gives $\perp$ for the suffix $\pi' = (P_{q+1}, P_q, \ldots, P_1, P_i)$ of π. So P_{q+1} must relay x (with any grade).

Also note, that P_q and P_{q+2} will both send their messages to P_{q+1} in round $r + 1$. Because $\pi \subseteq \mathsf{Conf}_s$, and in particular $P_{q+2} \in \mathsf{Conf}_s$, P_{q+1} cannot accept a message from P_{q+2}, unless it also states to have received x from P_k in round r. Thus P_{q+2} can only send x along the path $P_k, P_{q+2}, P_{q+1}, ..., P_1, P_i$.

However, in round $r + s + 1$, P_q itself receives (from P_{q-1}) the proof (as part of $\mathsf{I}_j^{(r)}$) that P_{q+1}'s true grade for P_k is 1. If P_{q+1} had relayed grade 0 instead, P_q now holds two conflicting P_{q+1}-signed claims for the same type and round, and `checkConsistencyParty`'s first check forces P_q to accuse P_{q+1}, collapsing that suffix of π to $\perp$. So we may assume P_{q+1} in fact relayed $(x, 1, \pi \subseteq \mathsf{Conf}_{s+1})$, and the window advances to start at P_{q+1}.

Iterating for $q = 1, \ldots, c - 2$ shows that either every party on π relays x throughout, giving $\mathsf{I}_i^{(r+c-2)}[\pi].\mathsf{I} = x = m_k$, or some party's lie about its grade is caught along the way, collapsing the corresponding suffix of π to $\perp$. This proves the lemma. □

Lemma 43 only covers a single augmented hop: the one immediately feeding into P_j. In general P_j's reference path π can be arbitrarily long, and `pathAugmentation` may insert a cluster of parties at *every* hop of π at once, not just the last one – each hop's cluster is constructed independently, per Protocol `pathAugmentation`. The next lemma shows the same guarantee survives this: regardless of how many of π's hops are augmented, and by how much, the value P_i ends up with at the end of the fully augmented path π' still agrees with the value P_j computes by walking the original, unaugmented π.

Lemma 44. *Let P_i, P_j be honest and alive parties and let P_k be a third party. Assume that in round $r \leq t+1$, P_j observes the message m_k from P_k to be valid (and correctly announced). For any path π ending in (P_k, P_j), and any path π' ending in P_i such that (π', P_j) is a path augmentation of π, we have for round $r' = r + (len(\pi') - len(\pi))$, where P_i is alive, that $\mathsf{I}_i^{(r')}[\pi'].\mathsf{I} = \mathsf{I}_j^{(r)}[\pi].\mathsf{I}$ or $\mathsf{I}_i^{(r')}[\pi''] = \perp$ for some suffix π'' of π'.*

Proof. Note that we care for round $r' = r + (len(\pi') - len(\pi))$, as we are interested in the message that arrives after having included all parties new to the path thanks to the augmentation.
We can then prove this lemma by induction on the number of hops of π.

Base case. If π has a single hop, $\pi = (P_k, P_j)$, then the claim is exactly Lemma 43.

Inductive step. Write $\pi = (\pi_h, P_j)$, where π_h is π's prefix ending in P_k, with h hops. Since P_j accepts m_k as valid, `checkValidity`'s own recursive check on m_k guarantees that P_k itself validly checked whatever it claims to have received from π_h's own last hop, so the induction hypothesis applies to π_h, with P_k playing the role P_j played before: for any augmentation π'_h of π_h, ending at some party $P_{i'}$, we have $\mathsf{I}_{i'}^{(q)}[\pi'_h].\mathsf{I} = \mathsf{I}_{P_k}^{(\cdot)}[\pi_h].\mathsf{I}$, or $\perp$ for some suffix, where q is the round at which π'_h's augmentation resolves.

Separately, since P_j observes m_k from P_k to be valid, Lemma 43 applies directly to the outermost hop (P_k, P_j): for any augmentation of this hop alone, any path from P_k to some party P_i, with c

nodes gives $\mathsf{I}_i^{(q+c-2)}[P_k,..,P_i].\mathsf{I} = \mathsf{I}_j^{(r)}[P_k].\mathsf{I}$, or $\perp$ for some suffix.

By Protocol `pathAugmentation`, the full augmentation π' of π is exactly π'_h followed by such a chain from P_k to P_i, since each hop of π is augmented independently. And $\mathsf{I}_{P_k}^{(\cdot)}[\pi_h].\mathsf{I}$ is precisely the value that $\mathsf{I}_j^{(r)}[P_k].\mathsf{I}$ nests one level inside, the same value $\mathsf{I}_j^{(r)}[\pi].\mathsf{I}$ recurses into by definition of the path-access operator. Chaining the two steps, first through π'_h, landing on P_k's observed value for π_h, then through the augmented final hop, landing on P_j's value for P_k, gives $\mathsf{I}_i^{(r')}[\pi'].\mathsf{I} = \mathsf{I}_j^{(r)}[\pi].\mathsf{I}$ at $r' = q + c - 2$, or $\perp$ for some suffix of π' if either step collapses. □

Value Agreement. With propagation established, we now show that value sets stabilize across honest parties, mirroring Section 5.3's own progression from propagation to value agreement.

The next lemma corresponds to Lemma 30 of Protocol Π_{base}. It captures the notion that a message path leading through an honest party P_i cannot be removed by an honest party P_j, unless P_j has already observed the value propagated along that path.

Lemma 45. *Let P_i, P_j be honest and alive parties. Assuming in round $r \leq t$ party P_i contains the path $\pi = (P_s,..,P_i)$ in $M_{acc-i}^{(r)}$, such that $len(\pi) = r+1$ and $P_j \notin \pi$. Then $\mathsf{I}_i^{(r)}[\pi].\mathsf{val} \in V_j^{(r)}$ or, for any round $r' \leq r+1$ it holds that $(\pi, P_j) \in M_{acc-j}^{(r')}$.*

Proof. We can observe that if $(\pi, P_j) \in M_{acc-j}^{(r+1)}$, then also $(\pi, P_j) \in M_{acc-j}^{(r')}$ for any $r' \leq r+1$. It thus suffices to check under what conditions the path could have been removed at some round $r' \leq r+1$. Since $\pi \in M_{acc-i}^{(r)}$, P_i has observed no accusation anywhere along π.

Case 1: P_j observed $\mathsf{I}_j^{(r')}[\pi'] = \perp$ for a suffix $\pi' = (P_1,..,P_i,P_j)$ of (π, P_j). By Lemma 38, P_j's recorded message from the honest P_i is identical to P_i's own, which would imply that $\mathsf{I}_i^{(r'-1)}[(P_1,..,P_i)] = \perp$ as well (by Lemma 42), which directly contradicts $\pi \in M_{acc-i}^{(r)}$.

Case 2: In round $r' \leq r+1$ P_j removed the path $(\pi, P_j) = (P_s,..,P_i,P_j)$ because it was a path-augmentation of $\pi' = (P_s,...,P_j)$ of length $r'+1$. That means, that in round r' P_j added the value $\mathsf{I}_j^{(r')}[\pi'].\mathsf{val}$ to $V_j^{(r')}$. Furthermore, by Lemma 44 and the fact that $\pi \in M_{acc-i}^{(r)}$, we know that $\mathsf{I}_i^{(r)}[\pi].\mathsf{I} = \mathsf{I}_j^{(r')}[\pi'].\mathsf{I}$, which implies that $\mathsf{I}_i^{(r)}[\pi].\mathsf{val} = \mathsf{I}_j^{(r')}[\pi'].\mathsf{val}$, so from round r', $\mathsf{I}_i^{(r)}[\pi].\mathsf{val} \in V_j^{(r')}$, and more specifically $\mathsf{I}_i^{(r)}[\pi].\mathsf{val} \in V_j^{(r)}$.

Case 3: In round $r' \leq r+1$ P_j removed the path (π, P_j) because its suffix $(\pi', P_j) = (P_1,..,P_i,P_j)$ is a path-augmentation of $\pi'' = (P_1,...,P_j)$ such that $\mathsf{I}_j^{(r')}[\pi''] = \perp$.

Since (π', P_j) genuinely augments π'' via `pathAugmentation`, P_j must validly observe π'''s final hop. Lemma 44's precondition is thus satisfied, and (using $\pi \in M_{acc-i}^{(r)}$ to rule out its $\perp$ alternative) it gives $\mathsf{I}_i^{(r'')}[\pi'] = \mathsf{I}_j^{(r')}[\pi'']$. But $\mathsf{I}_j^{(r')}[\pi''] = \perp$ by assumption, so $\mathsf{I}_i^{(r'')}[\pi'] = \perp$ too; since π' is a suffix of π, this directly contradicts $\pi \in M_{acc-i}^{(r)}$.

So Case 3 cannot occur. Together with Case 1 (also impossible), the only surviving possibility is Case 2: whenever (π, P_j) is removed from M_{acc-j}, it is because P_j has already added $\mathsf{I}_i^{(r)}[\pi].\mathsf{val}$ to $V_j^{(r)}$. □

The following lemma corresponds to Lemma 31 of Protocol Π_{base}, and proves that if a party P_i observes a value x signed by the sender for the first time in some round r, then honest and alive party P_j will observe that value x latest in the following round.

Lemma 46. *Let P_i, P_j be honest and alive parties. Let $r \leq t+1$ be the first round such that $x \in V_i^{(r)}$. Then, latest in round $r+1$ we have the guarantee that $x \in V_j^{(r+1)}$ and thus for any round $r' \geq r+1$ that P_j is alive it holds that $x \in V_j^{(r')}$.*

Proof. Let $r \leq t+1$ be the first round such that $x \in V_i^{(r)}$ and let $\pi \in M_{acc-i}^{(r)}$ be a path such that $len(\pi) = r+1$ and $\mathsf{I}_i^{(r)}[\pi].\mathsf{val} = x$. By Lemma 45 it is also satisfied that either $(\pi, P_j) \in M_{acc-j}^{(r+1)}$ or $x \in V_j^{(r)}$, assuming P_j is alive.

Thus, latest in round $r+1$ it holds that $x \in V_j^{(r+1)}$ (as $\mathsf{I}_j^{(r+1)}[(\pi, P_j)].\mathsf{val} = x$). Also, since $V_j^{(r)} \subseteq V_j^{(r+1)}$ (values are only added during the execution, never removed), it also holds that $x \in V_j^{(r')}$ for any $r' \geq r+1$ that P_j is alive. □

The following lemma corresponds to Lemma 32 of Protocol Π_{base}: we will prove that parties that have not terminated by round $f+2$ hold the same value sets.

Lemma 47. *Let P_i, P_j be honest parties alive in round $r = f+2$. Then $V_i^{(r')} = V_j^{(r')}$ for every $r' \geq r = f+2$.*

Proof. Let P_i, P_j be honest parties, that have not terminated by round $r = f+2$. Let $x \in V_i^{(r)}$. We will prove that this implies that $x \in V_j^{(r)}$ by case distinction on the round that x was added to $V_i^{(r)}$.

Case 1: Let $r' < f+2$ be the first round such that $x \in V_i^{(r')}$. By Lemma 46 this implies that by round $r'+1 \leq f+2$ we have $x \in V_j^{(r'+1)}$ and thus $x \in V_j^{(f+2)}$.

Case 2: Let $r' = f+2$ be the first round such that $x \in V_i^{(r')}$ and let π be the path such that $\pi \in M_i^{(r')}, len(\pi) = r'+1 = f+3$ and $\mathsf{I}_i^{(r')}[\pi].\mathsf{val} = x$. Given the length of π, there must be at least one other honest party P_k in π. This implies that there is a round $r'' < f+2$ such that $x \in V_k^{(r'')}$. By Lemma 46 this implies that by round $r''+1 \leq f+2$ we have $x \in V_j^{(r''+1)}$ and thus $x \in V_j^{(f+2)}$. □

The following lemma corresponds to Lemma 33 of Protocol Π_{base}. It serves to prove that once a party can terminate, because no more message paths are left, then the value sets for the other honest parties are fixed, even if they terminate multiple rounds apart.

Lemma 48. *Let P_i, P_j be honest parties. Let r be the first round s.t. there is no path $\pi \in M_i^{(r)}$ s.t. $len(\pi) \geq r+2$. Then, assuming P_j is alive, $V_j^{(r)} \subseteq V_i^{(r)}$ and $V_i^{(r)} = V_j^{(r')}$ for every $r' > r$.*

Proof. Let r be the first round s.t. there is no path $\pi \in M_i^{(r)}$ s.t. $len(\pi) \geq r+2$. By the termination check of Protocol Π_{fast}, this means P_i terminates in round $r+1$.

Assume by contradiction that in some round $r' \geq r$, honest party P_j adds a new value x to $V_j^{(r')}$, where $x \notin V_i^{(r)}$. This means, there is a path π^* such that $\pi^* \in M_j^{(r')}$, $len(\pi^*) = r'+1$ and $\mathsf{I}_j^{(r')}[\pi^*].\mathsf{val} = x$. Furthermore, it is clear that $P_i \notin \pi^*$, as otherwise P_i would have observed and accepted the value x, giving $x \in V_i^{(r)}$.

If $r' \leq t$, Lemma 45 (applied with P_j playing the role of the path-holding party, and P_i the other honest party) gives: $x \in V_i^{(r')}$, or for any round $r'' \leq r'+1$, $(\pi^*, P_i) \in M_{acc-i}^{(r'')}$. For $r'' = r$, the second option gives $(\pi^*, P_i) \in M_{acc-i}^{(r)}$: a path of length $r'+2$, strictly greater than $r+1$, contradicting that no path of length $\geq r+2$ remains there. Otherwise P_i must have already observed the value x, which would contradict $x \notin V_i^{(r)}$.

If $r' = t+1$, then there must be an honest party $P_k \in \pi^*$ (besides P_j), since a path of this length cannot consist entirely of corrupted parties. Applying Lemma 45 to the shortened prefix of π^* up to P_k in place of π^*, the same argument as above yields a contradiction.

This implies that $V_j^{(r')} \subseteq V_i^{(r)} \, \forall r' \geq r$. Thus, it remains to show that $V_i^{(r)} \subseteq V_j^{(r+1)}$, which follows directly from Lemma 46. Thus, if some party P_i no longer considers any message paths in some round r, then from the following round on, the value-sets of all honest parties are fixed. □

Path Augmentations. So far, our lemmas establish that Π_{fast} behaves exactly like Π_{base} whenever path augmentation is not involved. To actually bound the round complexity by the number of *actual* corruptions f rather than the worst case t, we need one more, purely combinatorial, ingredient: a guarantee that any sufficiently long path reduces to a short one via augmentation, regardless of how the adversary behaves.

In the following lemma we will prove a general fact about path augmentations: that 3 consecutive honest parties along a path will always be an augmentation of a shorter path.

Lemma 49. *Let π be any path $\pi = (\pi_1, P_i, P_j, P_k, \pi_2)$ that contains 3 consecutive honest parties $P_i \neq P_s, P_j, P_k$. Then π is a path augmentation of the path $\pi' = (\pi_1, P_i, P_k, \pi_2)$.*

Proof. Since P_i is honest, validity of Definition 54 guarantees that every honest party and in particular P_j and P_k, receive P_i's announcement with grade 1. Furthermore, validity also guarantees that every pair of honest parties has the other in its confirmation set for that announcement, each entry accompanied by a valid signature on P_i's true value. So instantiating $C = \{P_i, P_j, P_k\}$ for the hop (P_i, P_k) of π' in Protocol `pathAugmentation`: every $P \in C$ satisfies $\mathsf{WB}_P^{(r)}[P_i] = I_k^{(r+1)}[P_i]$ matching P_i's true value, $\mathsf{G}_P^{(r)}[P_i] = 1$, and $C \subseteq \mathsf{Conf}_P^{(r)}[P_i]$. Thus $\pi = (\pi_1, P_i, P_j, P_k, \pi_2)$ is a path augmentation of $\pi' = (\pi_1, P_i, P_k, \pi_2)$. □

Lastly, we prove that any path of length $\geq 3f + 4$ is an augmentation of a path of length $\leq 3f + 3$.

Lemma 50. *Let π be any path of length $\geq 3f + 4$. Then there is a path π^* such that π is a path augmentation of π^* and $len(\pi^*) \leq 3f + 3$.*

Proof. Let π be any path of length $\geq 3f + 4$. Along π there are at most f corrupted parties, splitting it into at most $f + 1$ runs of consecutive honest parties. Since π has at least $(3f + 4) - f = 2f + 4$ honest parties in total, at least 3 consecutive honest parties $P_i \neq P_s, P_j, P_k$ exist on π, i.e. $\pi = (\pi_1, P_i, P_j, P_k, \pi_2)$. By Lemma 49, π is a path augmentation of the one-shorter path $\pi' = (\pi_1, P_i, P_k, \pi_2)$. We can continue repeating this argument as long as the length of the one-shorter path $\pi' \geq 3f + 4$. Doing this step repeatedly, we obtain a path π^*, of length $\leq 3f + 3$, such that π is a path augmentation of π^*. □

Termination, Validity, Agreement. We now prove the following lemmas.

Lemma 51. *Protocol Π_{fast} achieves early termination.*

Proof. To prove that the protocol achieves early termination, we analyze an honest and alive party in round $r = 3f + 3$. Assume neither of the termination conditions applied. That would imply that there is a path $\pi \in M_i^{(r-1)}$ of length $\geq r + 1 = 3f + 4$. By Lemma 50, π is a path augmentation of a path π^* with $len(\pi^*) \leq 3f + 3 \leq r$. Since π^* is short, it must have been processed by round $len(\pi^*) \leq r - 1$: either π^* carried an observed accusation at that round, in which case π (being its augmentation) would have been removed by the protocol's first pruning pass (line 18); or π^* resolved to an actual value added to V_i, in which case π would have been removed by the second pruning pass (line 23). Either way, π would no longer be present in $M_i^{(r-1)}$.

Either case presents a contradiction to the assumption that an honest and alive party in round $r = 3f + 3$ has a path $\pi \in M_i^{(r-1)}$ of length $\geq r + 1 = 3f + 4$. Thus, any party will terminate latest in round $3f + 3$. Note that round 1 takes a single communication round, since the sender's initial broadcast requires no prior announcement phase. Each subsequent round takes 3 communication rounds due to the announcement-approach. Thus, any party will terminate latest after $1 + 3 \cdot (3f + 2) = 9f + 7$ communication rounds. □

Lemma 52. *Protocol* Π_{fast} *achieves validity.*

Proof. Assuming the sender P_s is honest, it sends $(0, x)$, correctly signed, to all parties in round 1 (Protocol Π_{fast}). Every honest party P_i thus sets $\mathsf{I}_i^{(1)}[P_s] = (0, x)$ and $V_i^{(1)} = \{x\}$.

Since P_s never signs a different round-1 value, and `checkValidity`'s recursive checks force every party's history to trace back consistently to a message signed by the sender, no honest party will witness another value. By Lemma 51, every honest party terminates, and thus does so on x. □

Lemma 53. *Protocol* Π_{fast} *achieves agreement.*

Proof. Let P_i, P_j be two honest parties, and let P_i be the party that terminates first (or they terminate in the same round). We will prove agreement by case distinction on the output of P_i and when the parties terminate.

Case 0: Assume P_i terminates in round $r \geq f + 2$. Given that P_i terminates first, we know that the other honest party P_j also terminates in round $\geq f + 2$. Applying Lemma 47 we know that $V_i^{(r')} = V_j^{(r')} \forall r' \geq r$ and thus both parties will terminate on the same value.

Case 1: Assume P_i terminates in round $r \leq f + 1$ on $\perp$ because for the first time $|V_i^{(r-1)}| > 1$. If $|V_j^{(r-1)}| > 1$, then P_j also terminates on $\perp$ in round r. Else, applying Lemma 46 to $V_i^{(r-1)}$ shows that any element $x \in V_i^{(r-1)}$ is also in $V_j^{(r)}$, so $|V_j^{(r)}| > 1$ and P_j terminates on $\perp$ (one round after P_i), matching P_i's output.

Case 2: Assume P_i terminates in round $r \leq f + 1$ because for the first time there is no path of length $\geq r+1$ in $M_i^{(r-1)}$. Then by Lemma 48 we know that $V_j^{(r-1)} \subseteq V_i^{(r-1)} = V_j^{(r')}$ for every $r' \geq r$. Thus, if P_j terminates in round $> r$, then P_i and P_j will have equal outputs, as $V_i^{(r-1)} = V_j^{(r')}$ for the round $r' \geq r$ at which P_j terminates. If P_j terminates in the same round r, we can apply Lemma 48 again (with the roles of P_i, P_j swapped) to get $V_i^{(r-1)} \subseteq V_j^{(r-1)}$, which together with the first inclusion gives $V_j^{(r-1)} = V_i^{(r-1)}$ and they agree on their outputs. Note that P_j cannot terminate on $\perp$ in round r, as that would contradict $V_j^{(r-1)} \subseteq V_i^{(r-1)}$. □

Theorem 37. *Assuming a plain PKI, Protocol* Π_{fast} *is a t-secure broadcast protocol that runs in* $9f + 7$ *rounds.*

Proof. This follows from Lemma 51, Lemma 52 and Lemma 53. □

References

[1] Ittai Abraham, T.-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In Peter Robinson and Faith Ellen, editors, *38th ACM PODC*, pages 317–326. ACM, July / August 2019.

[2] Ittai Abraham and Danny Dolev. Byzantine agreement with optimal early stopping, optimal resilience and polynomial complexity. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, page 605–614, New York, NY, USA, 2015. Association for Computing Machinery.

[3] Andreea B. Alexandru, Julian Loss, Charalampos Papamanthou, and Giorgos Tsimos. Sublinear-round broadcast without trusted setup against dishonest majority. Cryptology ePrint Archive, Report 2022/1383, 2022.

[4] Piotr Berman and Juan A Garay. Cloture votes: n/4-resilient distributed consensus in t+1 rounds. *Mathematical systems theory*, 26:3–19, 1993.

[5] Piotr Berman, Juan A Garay, and Kenneth J Perry. Optimal early stopping in distributed consensus. In *Distributed Algorithms: 6th International Workshop, WDAG'92 Haifa, Israel, November 2–4, 1992 Proceedings 6*, pages 221–237. Springer, 1992.

[6] Elette Boyle, Ran Cohen, and Aarushi Goel. Breaking the $O(\sqrt{(}n))$-bit barrier: Byzantine agreement with polylog bits per party. In Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen, editors, *40th ACM PODC*, pages 319–330. ACM, July 2021.

[7] Wonseok Choi, Wookjin Choi, Xiangyu Liu, and Vassilis Zikas. On the round complexity of early stopping. Cryptology ePrint Archive, Paper 2026/1878, 2026.

[8] Brian A Coan. Efficient agreement using fault diagnosis. *Distributed computing*, 7:87–98, 1993.

[9] Giovanni Deligios, Ivana Klasovita, and Chen-Da Liu-Zhang. Asymptotically optimal early termination for dishonest majority broadcast. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part V*, volume 15605 of *LNCS*, pages 116–144. Springer, Cham, May 2025.

[10] Danny Dolev. Unanimity in an unknown and unreliable environment. In *22nd Annual Symposium on Foundations of Computer Science (sfcs 1981)*, pages 159–168. IEEE, 1981.

[11] Danny Dolev. The byzantine generals strike again. *J. Algorithms*, 3(1):14–30, 1982.

[12] Danny Dolev, Ruediger Reischuk, and H Raymond Strong. 'eventual'is earlier than'immediate'. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 196–203. IEEE, 1982.

[13] Danny Dolev, Ruediger Reischuk, and H Raymond Strong. Early stopping in byzantine agreement. *Journal of the ACM (JACM)*, 37(4):720–741, 1990.

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

[15] Fatima Elsheimy, Simon Holmgaard Kamp, and Julian Loss. Early stopping is cheap. Cryptology ePrint Archive, Paper 2025/1107, 2025.

[16] Fatima Elsheimy, Simon Holmgaard Kamp, Julian Loss, and Charalampos Papamanthou. Near-optimal early stopping with optimal communication. Cryptology ePrint Archive, Paper 2026/2553, 2026.

[17] Fatima Elsheimy, Julian Loss, and Charalampos Papamanthou. Early stopping byzantine agreement in $(1+\epsilon)\cdot f$ rounds. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part VI*, volume 15489 of *LNCS*, pages 398–424. Springer, Singapore, December 2024.

[18] Fatima Elsheimy, Julian Loss, and Charalampos Papamanthou. Towards optimal early stopping agreement protocols. Cryptology ePrint Archive, Paper 2025/246, 2025.

[19] Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *20th ACM STOC*, pages 148–161. ACM Press, May 1988.

[20] Matthias Fitzi, Chen-Da Liu-Zhang, and Julian Loss. A new way to achieve round-efficient byzantine agreement. In Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen, editors, *40th ACM PODC*, pages 355–362. ACM, July 2021.

[21] Juan A Garay and Yoram Moses. Fully polynomial byzantine agreement for n> 3 t processors in t+ 1 rounds. *SIAM Journal on Computing*, 27(1):247–290, 1998.

[22] Diana Ghinea, Vipul Goyal, and Chen-Da Liu-Zhang. Round-optimal byzantine agreement. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 96–119. Springer, Cham, May / June 2022.

[23] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 445–462. Springer, Berlin, Heidelberg, August 2006.

[24] Leslie Lamport and Michael Fischer. Byzantine generals and transaction commit protocols. *Unpublished*, April 1982.

[25] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, jul 1982.

[26] Julian Loss and Jesper Buus Nielsen. Early stopping for any number of corruptions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part III*, volume 14653 of *LNCS*, pages 457–488. Springer, Cham, May 2024.

[27] Silvio Micali. Very simple and efficient byzantine agreement. In Christos H. Papadimitriou, editor, *ITCS 2017*, volume 4266, pages 6:1–6:1, 67, January 2017. LIPIcs.

[28] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[29] Kenneth J Perry and Sam Toueg. An authenticated byzantine generals algorithm with early stopping. Technical report, Cornell University, 1984.

[30] Birgit Pfitzmann and Michael Waidner. Unconditional byzantine agreement for any number of faulty processors. In *STACS 92: 9th Annual Symposium on Theoretical Aspects of Computer Science Cachan, France, February 13–15, 1992 Proceedings 9*, pages 337–350. Springer, 1992.

[31] Rüdiger Reischuk. A new solution for the byzantine generals problem. *Information and Control*, 64(1-3):23–42, 1985.

[32] Shravan Srinivasan, Julian Loss, Giulio Malavolta, Kartik Nayak, Charalampos Papamanthou, and Sri Aravinda Krishnan Thyagarajan. Transparent batchable time-lock puzzles and applications to byzantine consensus. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 554–584. Springer, Cham, May 2023.

[33] Sam Toueg, Kenneth J Perry, and TK Srikanth. Fast distributed agreement. *SIAM Journal on Computing*, 16(3):445–457, 1987.

[34] Jun Wan, Hanshen Xiao, Srinivas Devadas, and Elaine Shi. Round-efficient byzantine broadcast under strongly adaptive and majority corruptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 412–456. Springer, Cham, November 2020.

[35] Jun Wan, Hanshen Xiao, Elaine Shi, and Srinivas Devadas. Expected constant round byzantine broadcast under dishonest majority. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 381–411. Springer, Cham, November 2020.

Appendix

A Confirming Weak Broadcast

Path augmentation (Section 1.3) requires more than the plain information vector I can certify: a party must be able to prove not just what it received, but that another party independently witnessed the exact same announcement, something I alone cannot certify. *Confirming Weak Broadcast* is the primitive Π_{fast} uses each round to supply this: beyond the usual weak-broadcast guarantees of validity for an honest sender and consistency among parties that grade an announcement highly, it also lets a party who grades an announcement highly walk away with a non-repudiable, signed set of everyone else who confirmed the identical value. We first give the definition, then present one instantiation achieving it.

Definition 54 (Confirming Weak Broadcast)**.** Let Π be a protocol executed among parties $P_1, ..., P_n$, where a designated sender P_s holds an input v and all parties P_i output a value y_i, a grade $g_i \in \{0, 1\}$, and a set C_i of (party, signature) pairs, upon terminating. A protocol Π achieves *Confirming Weak 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 $y_i = v$ and $g_i = 1$. Additionally, for each pair of honest parties P_i and P_j, C_i contains (P_j, σ_j) where σ_j is a valid signature by P_j on v.
- Weak Consistency: If honest P_i and honest P_j both have $g_i = g_j = 1$, then $y_i = y_j$.
- Confirmation: If an honest party P_i has $g_i = 0$, then $\mathsf{C}_i = \varnothing$. Otherwise, for every honest P_i, C_i contains (P_s, σ_s) where σ_s is a valid signature by P_s on y_i and for every $(P_j, \sigma_j) \in \mathsf{C}_i$: σ_j is a valid signature by P_j on y_i. If additionally P_j is honest, then $y_j = y_i$.
- Termination: Every honest party terminates.

One instantiation achieving Definition 54 is the following two-round protocol.

Protocol `Confirming Weak Broadcast`

Assume sender P_s has input x.

Party P_s in round $(r = 1)$:

1: P_s sends $(x, \sigma_s = \mathsf{Sgn}_{sk_s}(x))$ to all parties.

Party $P_i \in \mathcal{P}$ does the following:

Round 1:
1: Initialize: $g_i = 0, y_i = \perp, \mathsf{C}_i = \varnothing$.
2: Let (x', σ_s) be a message received from P_s:
3: If σ_s is a valid signature by P_s on x', set $y_i = x'$; else set $y_i = \perp, \sigma_s = \perp$.
Round 2:
4: If $y_i \neq \perp$, send $(y_i, \sigma_s, \sigma_i = \mathsf{Sgn}_{sk_i}(y_i))$ to all parties.
5: Let $(y'_1, \ldots, y'_n)$ be the values from received messages $(y'_j, \sigma_s^{(j)}, \sigma_j)$ such that $\sigma_s^{(j)}$ verifies as P_s's signature on y'_j and σ_j verifies as P_j's signature on y'_j; else set $y'_j = \perp$.
6: If $y'_j \in \{\perp, y_i\}$ for all P_j, and $y_i \neq \perp$, set $g_i = 1$.
7: If $g_i = 1$, set $\mathsf{C}_i = \{(P_j, \sigma_j) \mid P_j \text{ where } y'_j = y_i\} \cup \{(P_s, \sigma_s)\}$.
8: Return (y_i, g_i, C_i).

Proof of Correctness

We now show that Protocol `Confirming Weak Broadcast` achieves each property of Definition 54 in turn.

Lemma 55. *Protocol* `Confirming Weak Broadcast` *achieves validity.*

Proof. Since P_s is honest, in round 1 it sends x, correctly signed, to all parties, so every honest party sets $y_i = x$.

In round 2, every honest party forwards x together with P_s's round-1 signature, additionally signed by itself. Since P_s is honest, it never signs any value other than x in round 1, so no party can produce a message that is validly signed by both P_s and itself for any value other than x. Hence every honest party P_i will in round 2 receive values that are either $\perp$ or x, satisfying the grade condition; combined with $y_i = x \neq \perp$, every honest party sets $g_i = 1$. Additionally, since every honest party forwarded the message of P_s, every honest party together with its signature on x is included in C_i. □

Lemma 56. *Protocol* `Confirming Weak Broadcast` *achieves weak consistency.*

Proof. Assume by contradiction that honest P_i, P_j have $g_i = g_j = 1$ but $y_i \neq y_j$. Since $g_j = 1$, the protocol's grade condition requires $y_j \neq \perp$. In round 2, since P_i and P_j are both honest, P_i receives P_j's true value: $y'_j = y_j$ (where y'_j denotes what P_i received from P_j). Since $y_j \neq \perp$ and $y_j \neq y_i$ (by assumption), we have $y'_j \notin \{\perp, y_i\}$, so P_i's own grade condition fails and $g_i = 0$, contradicting $g_i = 1$. □

Lemma 57. *Protocol* `Confirming Weak Broadcast` *achieves confirmation.*

Proof. If P_i has $g_i = 0$, then $\mathsf{C}_i = \varnothing$ by initialization, since it is only ever reassigned in the $g_i = 1$ branch.

If P_i has $g_i = 1$, then y_i was set to a non-$\perp$ value x' in round 1 upon receiving x' with a matching valid signature σ_s from P_s. The pair (P_s, σ_s) is included in C_i by construction. Additionally, for each P_j where $(P_j, \sigma_j) \in \mathsf{C}_i$, then by construction we have $y'_j = y_i$, where y'_j is the round-2 message P_i received from P_j and σ_j is a valid signature by P_j on $y'_j = y_i$. If additionally P_j is honest, it truthfully forwarded its own value y_j, so $y'_j = y_j$, giving $y_j = y_i$. □

Lemma 58. *Protocol* `Confirming Weak Broadcast` *achieves termination.*

Proof. Every honest party terminates in round 2 by construction. □