Optimal Early Termination for Dishonest Majority Broadcast

Giovanni Deligios[1], Ivana Klasovita[2], and Chen-Da Liu-Zhang[3]

[1] gdeligios@ethz.ch, ETH Zurich
[2] ivanakl@ethz.ch, ETH Zurich
[3] chen-da.liuzhang@hslu.ch, Lucerne University of Applied Sciences and Arts & Web3 Foundation

Abstract. Deterministic broadcast protocols among n parties tolerating t corruptions require $\min\{f+2, t+1\}$ rounds, where $f \leq t$ is the actual number of corruptions in an execution of the protocol. We provide the first protocol which is optimally resilient, adaptively secure, and asymptotically matches this lower bound for any $t < (1-\varepsilon)n$. By contrast, the best known algorithm in this regime by Loss and Nielsen (EUROCRYPT'24) always requires $O(\min\{f^2, t\})$ rounds. Our main technical tool is a generalization of the notion of polarizer introduced by Loss and Nielsen, which allows parties to obtain transferable cryptographic evidence of missing messages with fewer rounds of interaction.

1 Introduction

1.1 Setting

A broadcast protocol allows a sender to transmit a message to all parties and guarantees that 1) by the time it terminates, parties who follow the protocol instructions (honest parties) agree on a common message (a guarantee often referred to as *consistency*, or *agreement*), and 2) if the sender is among the honest parties, then the agreed-upon value is the intended sender's message (mostly referred to as *validity*). These guarantees must hold even if a subset of parties (the corrupted parties) arbitrarily deviates from the protocol instructions.

Because broadcast protocols serve as basic building blocks for more advanced tasks like secure computation protocols, significant efforts have been devoted to improving their efficiency under a wide variety of assumptions. Arguably one of the most investigated metrics of efficiency in the literature is *round complexity*, which is the maximum number of successive rounds of information exchange required in any protocol execution.

It is known (and has been for more than four decades) that when all parties are required to terminate after the same number of rounds of interactions, then any deterministic broadcast protocol among n parties requires at least $t+1$ rounds of interaction [DS83], where t is the upper bound on the corruption threshold. This lower bound holds even when corrupted parties' behavior is limited to crashing at some point through the protocol execution [LF82] and optimal solutions are known for any $t < n$ [DS83]. However, if the simultaneous termination condition is relaxed, and parties are allowed to terminate in different rounds, then the only known lower bound states that any deterministic broadcast protocol requires at least $f+2$ rounds of interaction, where f is the *actual* number of parties deviating from the protocol in a specific execution [DRS90].

Protocols whose number of rounds is proportional to the actual number of corruptions are referred to as *early-stopping* (or *early termination*) protocols [BGP92]. Round-optimal solutions requiring $\min\{t+1, f+2\}$ rounds are known for $t < n/3$ [AD15] and asymptotically optimal solutions exist for $t < n/2$ [PT84]. Round-optimal solutions exist for all $t < n$ assuming corrupted parties are only allowed to crash [Per85] or withhold messages [Ezh87] instead of arbitrarily deviating from the protocol. Surprisingly, in the case of malicious misbehavior and in the dishonest majority

setting (for $t \geq n/2$), little progress was made in almost forty years. Only recently, Loss and Nielsen [LN24] have shown the first optimally resilient early-stopping protocol (tolerating any number $t < n$ of parties arbitrarily deviating from the protocol) under minimal assumptions. However, their construction requires $O(\min\{f^2, t\})$ rounds of interaction. In this setting, the gap between the best-known solution and the known lower bounds remained open.

1.2 Contributions

We present the first early-stopping broadcast protocol which tolerates any number $t < n$ of malicious corruptions and is asymptotically round optimal whenever $t < (1 - \varepsilon)n$ for any constant $\varepsilon > 0$, meaning that in this case, if f is the number of parties that *actually* deviate from the protocol in a specific execution, our protocol runs in $O(f)$ rounds. Our construction also matches the communication complexity of [LN24] as well as its round complexity whenever $n - t = o(n)$, and is therefore strictly better than state of the art in this regime. Some form of setup is provably necessary to achieve broadcast in the dishonest majority setting [LSP19]. Analogously to [LN24], our construction only relies on a plain PKI setup, requires no number theoretic assumptions, and is secure against strongly adaptive adversaries. Our contributions are summarized by the following theorem, which is proven in Section 3.1.

Theorem 1 (Informal). *For all $n \in \mathbb{N}$ and for all $t < n$, there exists a broadcast protocol Π_{bc} tolerating up to t corruptions and such that, if f is the actual number of corruptions in an execution of Π_{bc}:*

- *Π_{bc} runs in $O(\min\{f^2, t\})$ rounds;*
- *Π_{bc} runs in $O(f)$ rounds if $t < (1 - \varepsilon)n$ for some $\varepsilon > 0$.*

1.3 Technical Overview

In this section, we discuss the main technical challenges posed by early-stopping broadcast in the dishonest majority setting, previous solutions, and new ideas to overcome their limitations.

Dealing with Missing Messages. In any distributed protocol, convincing others that a message was not received is hard. While a missing message is proof to the recipient P_i of the sender's corruption, this proof is not *transferable*. A party P_i who does not receive a message from P_j might send an accusation message towards P_j to all parties, but others cannot verify if P_j is withholding messages or if P_i is falsely accusing P_i despite having received the message.

Transferable Evidence of Missing Messages. In honest majority broadcast (when $t < n/2$) the solution is simple: each party that does not receive a message from the sender in the first round signs an accusation message against the sender and forwards it to all parties. Because it is guaranteed that $n - t > t$, any set of such accusations (a *certificate*) must include one from an honest party, and is *transferable* evidence of the sender's corruption. If after the second round no such certificate exists, then at least one honest party has received the sender's message. Although ensuring agreement remains non-trivial, the problem explained in the previous paragraph is solved.

Unfortunately, in the dishonest-majority regime (when $t \geq n/2$), such counting arguments no longer work. A solution, introduced in [LN24], involves a weaker, context-dependent analog of a certificate called a *polarizer*. The idea is that a set of signed accusations from *all* parties in subset $\mathcal{A}$ towards *all* remaining parties in $\mathcal{C} = \{P_1, \ldots, P_n\} \setminus \mathcal{A}$ serves the purpose of a certificate *in any*

protocol that guarantees that honest parties never accuse each other. Party P_i accepts a polarizer as valid evidence of P^*'s corruption whenever $P_i \in \mathcal{A}$ and $P^* \in \mathcal{C}$. If some honest P_i accepts a polarizer, then any other honest P_j must also accept it, because if $P_j \in \mathcal{C}$, this would imply P_i accused P_j at some point in the protocol, and this is guaranteed not to happen. Therefore, analogously to certificates in the honest majority setting, polarizers can be used as transferable evidence of some party's corruption.

Certificates vs. Polarizers. Still, certificates and polarizers differ in two major ways. The first is that, when $t < n/2$, corrupted parties cannot produce $n - t$ valid signatures against an honest party (assuming honest parties do not accuse each other), but in the dishonest majority setting corrupted parties can collude to produce polarizers with all honest parties in $\mathcal{C}$ and corrupted ones in $\mathcal{A}$. From the perspective of an external observer, there is no way to know which set contains honest parties. However, honest parties in the protocol do not accept such a polarizer, so that *within the protocol's context* this creates no problem.

The second is that in the honest majority setting certificates against a sender withholding messages can be constructed in just two rounds, while in the dishonest majority setting it is not even clear how, in the same scenario, parties can produce a polarizer against the sender, let alone how many rounds of interaction this requires.

Building Polarizers. Solving the latter problem is where most of [LN24] heavy lifting is done, and in the following, we recap their construction. In the simplest terms, the goal is to design a protocol that guarantees that honest parties obtain either 1) a polarizer against the sender, or 2) some message with a valid signature from the sender.

The solution in [LN24] relies on digital signatures and works as follows. In the first round, the sender P^* is required to transmit their signed message to all parties. In the second round, as in the honest majority setting, each party who *does not* receive a signed message from P^* signs and sends an accusation against the sender to all parties. Vice versa, each party who *does* receive a signed message from P^* forwards it to all parties.

Consider the point of view of a party P_i who *does not* receive a message from P^* in the first round. In the second round, P_i expects every other party P_j to either forward a message from the sender or an accusation against the sender. If the sender happens to be the only corrupted party and does not send any messages in the first round, then after the second round P_i has accusations from all parties towards P^*: this is a valid polarizer against P^*. However, it might be that more parties are corrupted in the second round, and a corrupted party P_ℓ neither forwards a message from P^* nor an accusation against P^* to P_i. In this scenario, after the second round, P_i obtains neither a signed message from P^* nor a valid polarizer, as they are missing the accusation from P_ℓ to P^*. However, at this point, P_i knows that P_ℓ is corrupted. Therefore, in a third round of the protocol P_i signs and sends an accusation against P_ℓ to all parties. The protocol proceeds recursively in this fashion, by also *forwarding* all signed messages and accusations, until either a message from the sender or a valid polarizer against the sender is obtained.

While it is clear that honest parties never accuse each other, it is not obvious why, if f is the actual number of corruptions in an execution of this protocol, its round complexity is proportional to f, making it an early-stopping protocol. Again, we remark that this protocol alone still does not provide any agreement guarantees, but serves as a fundamental building block for early-stopping broadcast.

To see this, suppose that an honest party P_i is still running after round r. This means that they have *not* received a signed message from the sender, but also *not* obtained a polarizer against the

sender. The latter implies that there exists no set $\mathcal{C}$ with $P^* \in \mathcal{C}$ such that P_i has accusations from all parties in $\mathcal{A} = \{P_1, \ldots, P_n\} \setminus \mathcal{C}$ towards all parties in $\mathcal{C}$ (otherwise, these accusations would be considered a valid polarizer against the sender). Therefore, if $\mathcal{C}_1 = \{P^*\}$, there exists some $P_{i_2} \notin \mathcal{C}_1$ such that P_{i_2} *does not* accuse the sender in round 2. If $\mathcal{C}_2 = \mathcal{C}_1 \cup \{P_{i_2}\}$, again there exists a party $P_{i_3} \notin \mathcal{C}_2$ who *does not* accuse the sender in round 2 or *does not* accuse P_{i_2} in round 3. In this fashion, we can obtain a set $\mathcal{C}_r = \{P^*, P_{i_2}, \ldots, P_{i_r}\}$ such that P_{i_k} does not accuse at least one party in $\mathcal{C}_{k-1}$. Clearly, the sender P^* is corrupted, because they do not send a message to P_i in round 1. We can argue that party P_{i_2} is also corrupted: they do not accuse the sender but they do not forward any valid message from the sender in round 2 (or P_i would receive a valid message from the sender in round 3 and would not be running in round r). Simlarly, party P_{i_3} is corrupted, because they neither accuse P_{i_2} in round 3, nor forward an accusation from P_{i_2} against the sender in round 3. By this reasoning, we can show that all parties $\{P^*, P_{i_2}, \ldots, P_{i_r}\}$ are corrupted, and since there are at most f corruptions in this protocol execution, we conclude that $r \leq f$.

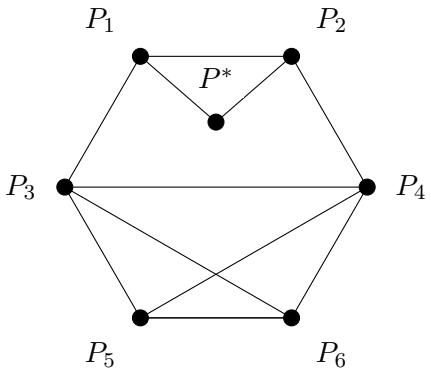
Building Polarizers, but Faster. The upper bound we have just shown on the number of rounds required to build polarizers à la [LN24] is tight: meaning an adversary corrupting f parties has a strategy that makes the protocol run for $f+1$ rounds. The strategy is simple: the sender is corrupted and does not send any messages. All corrupted parties follow the protocol instructions but the i-th corrupted party simply stops speaking in the i-th round. It is a good exercise to check understanding of the polarizer-building protocol in [LN24] to verify that an execution of their protocol against such an adversary requires $f + 1$ rounds.

We present a new, faster, polarizer-building protocol. We observe that the upper bound t on the number of corruptions is *never* used in [LN24]. By leveraging this upper bound, we can build polarizers in a *constant* number of rounds, whenever $t < (1 - \varepsilon)n$ for some constant $\varepsilon > 0$. This means that even the knowledge that a mere 0.01% of parties are guaranteed to be honest is enough to build polarizers in a number of rounds independent of f. Our protocol borrows techniques from the literature of dishonest-majority randomized broadcast protocols, and in particular from [WXSD20].

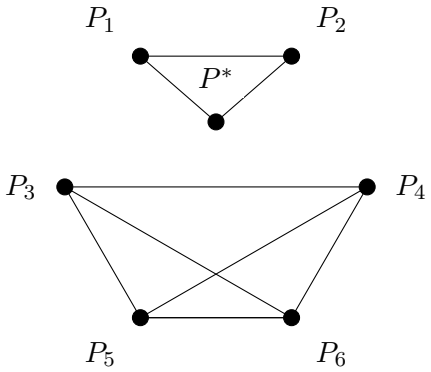
To begin, we observe that it is not necessary to collect signatures from *all* parties in $\mathcal{A}$ against *all* parties in $\mathcal{C}$ to ensure the transferability of polarizers. To aid intuition, consider a setting where a set of 7 parties $\mathcal{P} = \{P^*, P_2, P_3, P_4, P_5, P_6\}$ engages in the protocol explained above, and suppose that it is guaranteed that $t < 0.6n$, so that 3 parties are honest in any execution. Suppose that an honest party P_6 observes, at some point in the protocol execution, the following set Acc_i of accusations, where $\textsc{Acc}_{k\ell}$ denotes a signed accusation from P_k against P_ℓ:

$$\begin{aligned}\mathsf{Acc}_i = \{&\textsc{Acc}_{3*}, \textsc{Acc}_{32}, \textsc{Acc}_{4*}, \textsc{Acc}_{41}, \textsc{Acc}_{5*},\\ &\textsc{Acc}_{51}, \textsc{Acc}_{52}, \textsc{Acc}_{6*}, \textsc{Acc}_{61}, \textsc{Acc}_{62}\}.\end{aligned} \tag{1}$$

It can be checked that the set of accusations Acc_i is not enough to constitute a valid polarizer (for P_6) against P^* in the sense of [LN24], because while P_1 and P_2 have not accused P^*, the accusations $\textsc{Acc}_{42}$ and $\textsc{Acc}_{31}$ are missing. To visualize the situation better, we can consider the undirected graph whose set of nodes is $\mathcal{P}$ and where the edge $\{P_k, P_\ell\}$ is in the graph if and only if $\textsc{Acc}_{k,\ell} \notin \mathsf{Acc}_i$:



Because the protocol *ensures* that honest parties never accuse one another, honest nodes are guaranteed to be part of a clique of size (at least) $n - t = 7 - 4 = 3$ in the graph above. Therefore, edges that are *not* part of any clique of size 3 do not connect honest nodes. In our protocol, whenever party P_6 observes new accusations, they mentally build the graph as explained above and then repeatedly *remove any edge which is not part of a clique of size* 3 until no more such edges exist.[4] When performed on the graph above, this process yields the following graph.



If after this pruning process, P_6 is disconnected from P^* (as is the case in this example), they accept the original set of accusations[5] as a valid certificate of the sender's corruption. If another honest party (let's say P_5) receives the same set of accusations and builds the graph in the same way, then P_5 is also not connected to P^* in the resulting graph, because it is guaranteed that P_5 and P_6 are connected (they are part of the honest clique which has size 3, so that the edge between them is not removed), and a path from P_5 to P^* would immediately give one from P_6 to P^*, a contradiction. Therefore, our version of polarizers is also transferable within the protocol's context.

This example demonstrates that our notion of polarizer requires *fewer* accusations to achieve the same result. However, it is not immediately clear to what degree this affects the round complexity of the protocol. Our asymptotic quadratic-to-linear speed-up in the actual number of corruptions is due to the following observation.

If in round 1 the sender P^* fails to transmit a message to an honest P_i, then P_i accuses P^*. Therefore, in round 2, in a graph constructed as described above the distance (shortest path)

[4] Deciding if a certain graph contains a clique of a given size is an NP-complete problem. We present our protocol with this inefficient subroutine for simplicity, and discuss how to make the protocol efficient in Section 4.

[5] We also refer to such a set of accusation as a *polarizer*, even though it is not a polarizer in the sense of [LN24]. In Definition 9 and Remark 4 we explain how these notions are related.

between P_i and P^* is at least 2. Similarly, if P_i does not receive a message from P^* in round 2 but lacks sufficient accusations to disconnect them from the sender, they accuse all parties who did not accuse P^* in the first round. This means that after round 2, any party is either: 1) not in P_i's neighborhood or 2) at distance at least 2 from P^*. These two conditions ensure that the distance between P_i and P^* is at least 3. By extending this reasoning, if P_i has not received a valid message or enough accusations by round r to disconnect them from the sender, then the distance between P_i and P^* is at least r.

However, any edge in a graph built as described above is part of a clique of size at least $n - t$, and the diameter d of any such graph (the maximum distance between any two nodes) is at most $2n/(n - t)$. Thus, for $n - t = O(n)$ (for example $t = 0.99n$), the diameter d of such a graph is constant in n. By the reasoning above, after a *constant* number of rounds any honest P_i either receives a message from P^* or is disconnected from them by the accusations it has observed (after our pruning subroutine), and therefore obtains a valid polarizer against P^*.

From Polarizers to Broadcast. A polarizer-building protocol as described so far seems like a far cry from an early-stopping broadcast; in particular, it provides no agreement guarantees whatsoever. However, turning this protocol into a broadcast protocol with $O(f)$ rounds of overhead mostly involves well-established techniques: first, from a polarizer-building protocol, one obtains primitives providing increasingly strong agreement guarantees (weak-broadcast, 2-graded broadcast). Then these primitives are run in succession, each time with a new sender, following the classic king-phase paradigm [Rei85]. Some extra care is needed to avoid the running time of the protocol to be asymptotically $\Omega(t)$ in some parameter regimes. However, since this construction only uses the polarizer-building primitive in a black-box way, we do not duplicate the details here, and instead refer the interested reader to [LN24].

1.4 Related Work

We first present an overview of *deterministic* early-stopping broadcast protocols, which is the focus of our paper. Deterministic broadcast requires $\min\{f + 2, t + 1\}$ rounds of communication [DS83, DRS90].

Plain Model Protocols ($t < n/3$). A classic result of Lamport, Shostak, and Pease showed that any broadcast protocol in the plain model can tolerate at most $t < n/3$ corruptions [LSP19]. Most work on early-stopping broadcast has been done in this setting. The first early-stopping protocol with optimal resilience is due to Dolev, Reischuk, and Strong [DRS82], and requires $\min\{2f + 5, 2t + 3\}$ rounds. Reischuk later proposed a protocol running in $2f + 3$ rounds but with lower resilience of $t < n/6$ [Rei85]. Toueg, Perry, and Srikanth improved on [DRS82] with a protocol running in $\min\{2f + 4, 2t + 2\}$ rounds and optimal resilience [TPS87]. This result was further improved by Berman, Garay, and Perry who presented a protocol running in $\min\{f(1 + 1/d) + 5, t(1 + 1/d)\}$ for any constant d in [BGP92]. The first protocol tightly matching the lower bound of $\min\{f + 2, t + 1\}$ rounds again appears in [DRS82], but only tolerates $2t^2 + 3t + 5 < n$ corruptions [DRS82]. For round-optimal solutions, the failure resilience was improved by Dolev to $n > \max\{4t, 2t^2 - 2t + 2\}$ in [DRS90] and again to $n \geq \left\lceil \sqrt{t} \right\rceil \cdot \left\lfloor 4t + \sqrt{t} + 1 \right\rfloor$ by Coan [Coa93]. Garay and Moses presented the first protocol attaining optimal early termination with linear failure resilience of $t < n/8$ [GM98]. In [BGP92] the first optimally resilient protocol with optimal early termination was also presented, but at the expense of exponential message complexity. Finally, Abraham and Dolev showed the first optimal early-stopping protocol with optimal resilience and polynomial message complexity [AD15].

Protocols with Setup ($n/3 \leq t < n$). When $t \geq n/3$, the lower bound of [LF82] can be circumvented by assuming some form of setup. This usually translates into the possibility of authenticating messages, for example, through computationally secure signature schemes [DS83] or information-theoretically secure pseudosignatures [PW92]. In this setting, any number $t < n$ of corruptions can be tolerated, and much less work has been done on early-stopping protocols. Perry and Toueg presented a protocol running in $2f + 4$ rounds with suboptimal resilience of $t < n/2$ [PT84]. Loss and Nielsen presented a broadcast protocol with optimal resilience of $t < n$ but suboptimal runtime of $O(\min\{f^2, t+1\})$ [LN24].

Randomized Protocols. In contrast to deterministic protocols, if parties have access to local randomness several works [FM88, KK06, Mic17, ACD+19, FLZL21] 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 failure probability $2^{-\lambda}$. This is improved to $\lambda^{-\lambda}$ failure probability in [GGLZ22]. Similarly, randomized protocols in the dishonest majority setting are known with an expected constant number of rounds [WXSD20] for $t = (1-\varepsilon)n$ for constant ε, and fixed number of rounds $O(\lambda)$ with $2^{-\lambda}$ probability of failure for $t = (1-\varepsilon)n$. The works [WXDS20, SLM+23, ALPT22] achieved higher levels of adaptive resilience assuming time-lock assumptions.

2 Preliminaries

2.1 Model

Synchronous Deterministic Distributed Protocols. Informally, a synchronous protocol is a distributed algorithm for a certain task in which communication among different processes happens in *rounds*. In practice, such algorithms can be implemented over sufficiently reliable point-to-point networks (meaning messages are delivered within some known time) and when the local clocks of different processes are sufficiently synchronized. A variety of definitions have been proposed and used in the literature to formally treat such objects. For the purposes of this work, we are only concerned with *deterministic* protocols, and the following simple definition suffices.

Definition 1 (Distributed Protocol). *An **deterministic distributed protocol** Π among n parties is an n-tuple of function families*

$$\left(\left\{\mathsf{NxtMsg}_1^{(k)}\right\}_{k=1}^{\ell}, \ldots, \left\{\mathsf{NxtMsg}_n^{(k)}\right\}_{k=1}^{\ell}; \left\{\mathsf{Out}_1^{(k)}\right\}_{k=1}^{\ell}, \ldots, \left\{\mathsf{Out}_n^{(k)}\right\}_{k=1}^{\ell}\right) \tag{2}$$

where for all $i \in [1,n]$ and all $k \in [1,\ell]$

- *The function $\mathsf{NxtMsg}_i^{(k)}$ is an efficiently computable function called the k-th next message function of player (or party) P_i,*
- *$\left(m_{i1}^{(k)}, \ldots, m_{in}^{(k)}\right) \leftarrow \mathsf{NxtMsg}_i^{(k)}\left(\textsc{View}_i^{(k-1)}\right)$,*
- *$y_i^{(k)} \leftarrow \mathsf{Out}_i^{(k)}\left(\textsc{View}_i^{(k)}\right)$ is called the output of player P_i in round k, and it is guaranteed that $y_i^{(\ell)} \neq \perp$.*[6]

where, if x_i is the input of party P_i, then $\textsc{View}_i^{(k-1)}$ is an inductively defined set, called the view of party P_i in rounds $[1, k-1]$:

[6] We denote by $\perp$ some fixed value in the codomain of function $\mathsf{Out}_i^{(k)}$.

– $\textsc{View}_i^{(0)} = x_i$,
– *For all* $k \in [1, \ell]$ *we have* $\textsc{View}_i^{(k)} = \textsc{View}_i^{(k-1)} \cup \left\{m_{1i}^{(k)}, \ldots, m_{ni}^{(k)}\right\}$.

Protocol Setup. Broadcast protocols tolerating a dishonest majority of parties provably require some form of setup, meaning that parties have access to correlated randomness coming from a joint distribution which is independent of the protocol inputs. We write $\mathsf{SetUp} = (\mathsf{SetUp}_1, \ldots, \mathsf{SetUp}_n)$ to improperly denote both the distribution or one specific sample from the distribution. Most known protocols, including ours, adopt this setup in the form of a public key infrastructure (PKI) for digital signatures.[7]

Adversary. By *security* of a protocol, we informally mean that the protocol provides certain guarantees even when run in the presence of an adversarial entity that can corrupt a subset of the parties participating in the protocol, observe their state, and influence their behavior. A central adversary is used to model the scenario in which different dishonest parties collude and agree on a common strategy to disrupt the protocol execution. We deal with the strongest type of *efficient* (PPT, but the specific computational model is not relevant for the treatment of this paper) adversaries considered in the literature:

– *Active*, meaning that the adversary can make corrupted parties arbitrarily deviate from the protocol instructions, for example by withholding messages and/or sending wrong messages.
– *Rushing*, meaning that in each round, the adversary picks the corrupted parties' messages after having observed the messages sent by honest parties to corrupted parties in the same round.
– *Adaptive*,[8] meaning that the adversary can corrupt parties based on information gathered during the protocol execution.

To discuss security of a distributed protocol in a meaningful way, we must first define what it means to run a distributed protocol in the presence of such an adversary.

Definition 2 (Protocol Execution). *An* ***execution***

$$\textsc{Exec}_{\Pi}^{\mathcal{A}}(\mathbf{x}_{\mathsf{In}}, \mathsf{SetUp}) \tag{3}$$

of an n***-party protocol*** Π ***with setup*** $\mathsf{SetUp} = (\mathsf{SetUp}_1, \ldots, \mathsf{SetUp}_n)$ ***and inputs*** $\mathbf{x}_{\mathsf{In}} = (x_1, \ldots, x_n)$ ***against an efficient (***PPT***) active, rushing, adaptive adversary*** $\mathcal{A}$ *is a random vector defined via the following random experiment involving* $\mathcal{A}$ *and a challenger* $\mathcal{C}_{\Pi}$*. To begin,* $\mathcal{C}_{\Pi}$ *initializes:*

– *An empty set of corrupted indices* $\mathsf{Corrupt}^{(1)} \leftarrow \varnothing$,

[7] Broadcast protocols based on signatures, including ours, can provide information-theoretic security if a PKI is swapped for an appropriate setup for information-theoretic equivalents of signatures often referred to as pseudosignatures.

[8] We consider the strongest type of adaptive adversary, often referred to in the literature as *strongly adaptive*. This type of adversary can observe messages from all honest P_i to all corrupt P_j in round k, based on these messages decide to corrupt a subset $\mathcal{S}$ of the honest parties, and then change the messages from the (now corrupted) parties in $\mathcal{S}$ towards the remaining honest parties *in the same round* k. This is in contrast to what is known in the literature as a *weakly adaptive* adversary, that is not allowed to perform this type of message deletion attack. Restricting the behavior of adaptive adversaries in this manner is equivalent to assuming an *atomic send* primitive, which allows a party to input all its round messages and deliver all of them simultaneously to all parties.

– *A set of currently running parties* $\mathsf{Alive}^{(1)} = [1, n]$.
– *The view of party* P_i *as* $\textsc{View}_i^{(0)} \leftarrow (x_i, \mathsf{SetUp}_i)$ *for all* $i \in [1, n]$.

For all $k \in [1, \ell]$ *let* $\mathsf{Honest}^{(k)} = [1, n] \setminus \mathsf{Corrupt}^{(k)}$ *The experiment proceeds in rounds, where in each round* $k \in [1, \ell]$ *the following interaction takes place:*

1. *For all indices* $i \in \mathsf{Honest}^{(k)} \cap \mathsf{Alive}^{(k)}$ *the challenger* $\mathcal{C}_{\Pi}$ *computes*

$$\left(m_{i1}^{(k)}, \ldots, m_{in}^{(k)}\right) \leftarrow \mathsf{NxtMsg}_i^{(k)}\left(\textsc{View}_i^{(k-1)}\right) \tag{4}$$

   and sends $m_{ij}^{(k)}$ *to* $\mathcal{A}$ *for all* $j \in [1, n]$.[9]
2. *Then* $\mathcal{C}_{\Pi}$ *and* $\mathcal{A}$ *repeat up to* $\left|\mathsf{Honest}^{(k)} \cap \mathsf{Alive}^{(k)}\right|$ *times:*
   (a) *The adversary* $\mathcal{A}$ *selects any index* $i \in \mathsf{Honest}^{(k)} \cap \mathsf{Alive}^{(k)}$ *and sends* $(\textsc{Corrupt}, i)$ *to* $\mathcal{C}_{\Pi}$*;*
   (b) *The challenger* $\mathcal{C}_{\Pi}$ *sends* $\textsc{View}_i^{(k-1)}$ *to* $\mathcal{A}$.
3. *The adversary* $\mathcal{A}$ *sends* $\left(\widehat{m}_{i1}^{(k)}, \ldots, \widehat{m}_{in}^{(k)}\right)$ *to* $\mathcal{C}_{\Pi}$ *for all*

$$i \in \mathsf{Corrupt}^{(k)} \cup \{i \mid (\textsc{Corrupt}, i) \text{ sent to } \mathcal{C}_{\Pi} \text{ in round } k\}. \tag{5}$$

At the end of each round k *the challenger* $\mathcal{C}_{\Pi}$ *updates:*

– $\mathsf{Corrupt}^{(k+1)} = \mathsf{Corrupt}^{(k)} \cup \{i \mid (\textsc{Corrupt}, i) \text{ received from } \mathcal{A} \text{ in round } k\}$.
– *The view*

$$\textsc{View}_i^{(k)} \leftarrow \textsc{View}_i^{(k-1)} \cup \left\{\tilde{m}_{1i}^{(k)}, \ldots, \tilde{m}_{ni}^{(k)}\right\} \tag{6}$$

  for all honest indices $i \in \mathsf{Honest}^{(k+1)}$*, where for all* $j \in [1, n]$ *we define*

$$\tilde{m}_{ji}^{(k)} = \begin{cases} \widehat{m}_{ji}^{(k)} & \text{if } j \in \mathsf{Corrupt}^{(k+1)}, \\ m_{ji}^{(k)} & \text{otherwise.} \end{cases} \tag{7}$$

– *The output* $y_i^{(k)} \leftarrow \mathsf{Out}_i^{(k)}\left(\textsc{View}_i^{(k)}\right)$ *for all honest indices* $i \in \mathsf{Honest}^{(k+1)} \cap \mathsf{Alive}^{(k)}$.
– $\mathsf{Alive}^{(k+1)} = \mathsf{Alive}^{(k)} \setminus \left\{i \in [1, n] \mid y_i^{(k)} \neq \perp\right\}$.

In this random experiment, the following random variables are defined:

– *The runtime of party* P_i *is*

$$\begin{aligned} r(P_i) &= \min\left\{k \in [1, \ell] \mid y_i^{(k)} \neq \perp\right\} \\ &= \max\left\{k \in [1, \ell] \mid i \in \mathsf{Alive}^{(k)}\right\}. \end{aligned} \tag{8}$$

– *The output of party* P_i *is* $y_i = y_i^{(r(P_i))}$.
– *The runtime of* Π *is defined as*

$$\begin{aligned} r(\Pi) &= \max\{r(P_i) \mid i \in [1, n] \setminus \mathsf{Corrupt}\} \\ &= \min\left\{k \in [1, \ell] \mid \mathsf{Alive}^{(k+1)} = \varnothing\right\}. \end{aligned} \tag{9}$$

[9] The adversary also receives all messages sent among honest parties to capture that channels between honest parties are authenticated, but not secure.

– *The number of corruptions is* $f = \left|\mathsf{Corrupt}^{(r(\Pi)+1)}\right|$.
– *The view* $\textsc{View}_{\mathcal{A}}$ *of the adversary is*

$$\left(\bigcup_{k=1}^{r(\Pi)} \left\{ \textsc{View}_i^{(k)} \mid i \in \mathsf{Honest}^{(k)} \wedge i \in \mathsf{Corrupt}^{(k+1)} \right\}, \right. \\ \left. \bigcup_{k=0}^{r(\Pi)} \left\{ m_{ij}^{(k)} \mid j \in [1,n] \wedge i \in \mathsf{Honest}^{(k+1)} \cap \mathsf{Alive}^{(k+1)} \right\} \right). \tag{10}$$

Finally, we set $\textsc{Exec}_{\mathcal{A}}^{\Pi}(\mathsf{SetUp}, x_{\mathsf{In}})$ *to be the random vector collecting all random variables defined by this random experiment.*

Remark 1. When dealing with adaptive adversaries the expression *honest party* is in general ambiguous, as the set of honest parties varies through the protocol execution. When not otherwise specified, we say that P_i is honest in execution $\textsc{Exec}_{\mathcal{A}}^{\Pi}(\mathsf{SetUp}, x_{\mathsf{In}})$ to mean that $i \in \mathsf{Honest}^{(r(\Pi))}$, or in other words that P_i is honest *throughout the whole protocol execution.*

Remark 2. When we say that a property has to hold in all executions of a protocol Π, we mean that it has to hold for $\textsc{Exec}_{\mathcal{A}}^{\Pi}(\mathsf{SetUp}, x_{\mathsf{In}})$ for all input vectors $\mathbf{x}_{\mathsf{In}}$, for all well-formed setups SetUp, and for all efficient (PPT) adversaries $\mathcal{A}$.

The next definition captures the notion of all values that an efficient adversary $\mathcal{B}$ *could* produce if they are given access to the view of adversary $\mathcal{A}$ in a given protocol execution, *and also* the outputs of honest parties in the same execution.

Definition 3 (Adversarial Value). *A value x is an* ***adversarial value*** *in* $\textsc{Exec}_{\mathcal{A}}^{\Pi}(\mathsf{SetUp}, x_{\mathsf{In}})$ *if and only if there exists an efficient (*PPT*) algorithm that given as inputs*

1. *The outputs of honest parties* $\{y_i\}_{i \in \mathsf{Honest}^{(r(\Pi))}}$*, and*
2. *The adversary view* $\textsc{View}_{\mathcal{A}}$*,*

outputs x with non-negligible probability. A value x is called an ***honest value*** *in* $\textsc{Exec}_{\mathcal{A}}^{\Pi}(\mathsf{SetUp}, x_{\mathsf{In}})$ *if* $x \in \textsc{View}^{(r(P_i))}$ *for some* $i \in \mathsf{Honest}^{(r(\Pi))}$*.*

Justifiers. A common way to define the security of a distributed protocol Π is to list predicates that must be satisfied by all honest outputs in any execution of Π. It is less clear how to define security properties that should hold *even for maliciously crafted outputs*, or why this is useful. The main problem is that without further restrictions, an adversary $\mathcal{A}$ can choose the outputs of corrupted parties arbitrarily (in fact, the notion of *output of a corrupted party* is not even defined), so that only trivial security properties can be achieved.

One way to solve this problem is to require parties to collect evidence during the execution that can be later provided as proof (a justifier) that a certain output was computed as prescribed by the protocol. Security properties are then required to hold for all outputs *with a valid proof.*

To explain why this is useful, consider an interactive protocol Π consisting of two successive subprotocols Π_1 and Π_2. Protocol Π instructs parties to take their output from Π_1 and use it as input to Π_2. Protocol Π_1 might provide some security guarantees that are needed for the security of Π_2, but with no restrictions in place, a corrupted party can provide an arbitrary input to Π_2. Justifiers allow to prevent this type of misbehavior: to provide input to Π_2 parties must attach

a valid proof certifying that this value was the output of Π_1, and the security guarantees of Π_1 hold for all outputs for which such a proof exists. This limits the ability of an adversary of undoing the progress achieved by subprotocol Π_1. We present the formal definition of justifier following the formalism of [LN24].

Definition 4 (t-Transferable Justifier). *Let Π be a distributed protocol among parties in $\mathcal{P}$. A **justifier predicate** for Π is an efficiently computable (PPT) predicate*

$$J : \mathcal{P} \times \{0,1\}^* \times \{0,1\}^* \to \{\texttt{true}, \texttt{false}\} \tag{11}$$

such that

- **(t-Transferability)** *For any execution of protocol Π with $f \leq t$ corruptions, and for all honest and adversarial values (m, π) with respect to this execution, it holds that if $J(P_i, m, \pi) = \texttt{true}$ for some honest $P_i \in \mathcal{P}$, then $J(P_j, m, \pi) = \texttt{true}$ for all honest $P_j \in \mathcal{P}$.*[10]

Remark 3. We write $J(m, \pi) = \texttt{true}$ as an abbreviation of $J(P_i, m, \pi) = \texttt{true}$ for all honest $P_i \in \mathcal{P}$.

Clearly, an adversary can try to produce fraudulent justifiers for arbitrary outputs, even after having observed the justifications for legitimate outputs of honest parties. This must be prevented from the protocol design. The following definition formally captures this notion of *fraudulent* output.

Definition 5 (Adversarial Justified Outputs). *Vector $\mathbf{w}_{\mathsf{Out}} = (w_1, \ldots, w_n)$ is called a vector of **adversarial justified outputs** with respect to execution $\textsc{Exec}^{\Pi}_{\mathcal{A}}(\mathsf{SetUp}, x_{\mathsf{In}})$ if and only if*

$$w_i = \begin{cases} y_i & \text{if } P_i \text{ is honest,} \\ \widehat{y}_i & \text{if } P_i \text{ is corrupted} \end{cases} \tag{12}$$

where $\widehat{y}_i$ is any adversarial value for $\textsc{Exec}^{\Pi}_{\mathcal{A}}(\mathsf{SetUp}, x_{\mathsf{In}})$ such that $\widehat{y}_i = (m, \pi)$ and $J(m, \pi) = \texttt{true}$, or $\widehat{y}_i = \bot$.

2.2 Primitives

Signatures. For the purposes of this work, a *signature scheme* Σ is a triple of efficient algorithms

$$\Sigma = (\mathsf{Kgn}, \mathsf{Sgn}, \mathsf{Vfy}) \tag{13}$$

with the following syntax:

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Kgn}()$
- $\sigma \leftarrow \mathsf{Sgn}_{\mathsf{sk}}(m)$
- $b \leftarrow \mathsf{Vfy}_{\mathsf{pk}}(m, \sigma)$

such that $\mathsf{Vfy}_{\mathsf{pk}}(m, \mathsf{Sgn}_{\mathsf{sk}}(m)) = \texttt{true}$ for all $m \in \{0,1\}^*$. Following the Dolev-Yao paradigm [DY83], we treat signatures as ideal objects in all of our protocols, meaning that we assume that an adversary simply cannot produce a signature σ such that $\mathsf{Vfy}_{\mathsf{pk}}(m, \sigma) = \texttt{true}$ unless they know the secret key sk. This is not entirely realistic, as regardless of how large the (finite) secret-key space is, an adversary can always guess sk with non-zero probability. Still, this approach is widely adopted in both the distributed computing and formal verification literature and in our specific case, it is possible to explicitly reduce the security of our constructions to the security of an underlying signature scheme. However, this introduces a significant overhead in the proofs and ultimately detracts from the clarity of the exposition.

[10] Transferability also holds for honest outputs in $\textsc{Exec}^{\Pi}_{\mathcal{A}}(\mathsf{SetUp}, x_{\mathsf{In}})$, since these are trivial adversarial values.

Send Transferable Message Protocol. The main focus of this work is to construct a round-efficient version of the *send transferable message* (STM) protocol in [LN24] (the polarizer-building protocol explained in Section 1.3). Such a protocol allows a designated party P^* (which we refer to as the *sender*) to send a message to multiple recipients so that each party P_j obtains, together with the message, a transferable proof of authenticity of the message. This could be easily achieved via digital signatures, except that we require such a proof to exist even if P^* is corrupted and does not send any message at all. Informally, the protocol provides the three guarantees below.

- If P^* is honest, all parties receive the intended sender's input (validity) and what is more, even dishonest recipients can only produce valid proofs for the intended sender's value.
- Any honest P_j obtains a valid proof of whatever message a dishonest sender decides to send them (justified outputs): this is true *even when the sender fails to send any message at all.* In this case, P_j obtains a proof that can convince any other honest party of the fact that P_i did not receive the intended message from P^*.
- Finally, parties expect the sender's message to satisfy certain properties specified by an appropriate predicate: only such messages will be accepted by honest parties, and messages failing to satisfy the predicate can be ignored (justified message). This is more than a purely syntactic check: for example proof that the input was produced as output from a different protocol.

These guarantees are formalized in the definition below.

Definition 6 (Send Transferable Message). *A triple $(\Pi, J_{\mathsf{In}}, J_{\mathsf{Out}})$ where Π is an interactive protocol with a single designated input-holding party P^*, and J_{In} and J_{Out} are justifier predicates for Π is a t-secure **send transferable message protocol** if the following properties hold in any execution of Π with $f \leq t$ corruptions:*

1. **(Justified Outputs)** *The output of every honest party P_i is of the form (m_i, π_i) and it holds that $J_{\mathsf{Out}}(P_i, m_i, \pi_i) = \mathtt{true}$.*
2. **(Justified Message)** *For all adversarial justified outputs $\mathbf{y} = (y_1, \ldots, y_n)$ with respect to J_{Out}, it holds that*
   - *$y_i = \perp$, or*
   - *$y_i = \textsc{NoMsg}$, or*
   - *$y_i = (m_i, \pi_i)$ with $\pi_i = \left(\pi_i^{\mathsf{In}}, \pi_i^{\mathsf{Out}}\right)$ and $J_{\mathsf{In}}\left(m_i, \pi_i^{\mathsf{In}}\right) = \mathtt{true}$.*
3. **(Validity)** *If P^* is honest with input m^*, for all adversarial justified outputs $\mathbf{y} = (y_1, \ldots, y_n)$ with respect to J_{Out} then*
   - *$y_i = \perp$, or*
   - *$y_i = (m^*, \pi)$.*

Broadcast. We now provide a formal the classical definition for early-stopping broadcast: termination is relaxed and not all honest parties are required to terminate in the same round.

Definition 7 (Broadcast). *An interactive protocol Π with a single designated input-holding party P^* is a t-secure **broadcast protocol** if the following properties hold in any execution of Π with $f \leq t$ corruptions:*

1. **(Validity)** *If P^* is honest with input m, then all honest parties output m.*
2. **(Agreement)** *All honest parties output the same value.*
3. **(Termination)** *Each honest party P_i terminates in a constant number of rounds, that is $r(P_i)$ is finite.*

3 Broadcast Protocol with Optimal Early Termination

3.1 Round Efficient Send Transferable Message Protocol

In this section, we present our round-efficient send transferable message protocol. We begin by presenting a generalization of the notion of *polarizer* introduced in [LN24]. Polarizers are the main technical tool that allows us to define sound justifier predicates for the outputs of our send transferable message protocol.

Generalized Polarizers. Before introducing the notion of a polarizer, we introduce some notation that will make the following definitions more compact. In the following, let $\Sigma = (\mathsf{Kgn}, \mathsf{Sgn}, \mathsf{Vfy})$ be a signature scheme. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be set of parties and pk_i be the public key of party P_i.

Definition 8 (Σ-accusation). *A Σ-**accusation** from party P_i against (or towards, or to) party P_j is a tuple*

$$((\textsc{Acc}, i, j), \sigma)\,. \tag{14}$$

It is called valid if and only if $\mathsf{Vfy}_{\mathsf{pk}_i}((\textsc{Acc}, i, j), \sigma) = \mathtt{true}$, and in such cases denoted by $\textsc{Acc}_{ij}$.

Polarizers can be understood as the equivalent of a digital signature on a "missing message". Similar to how a digital signature allows a P_i to prove to any other party P_j that they received a message m from a third party S, a polarizer allows P_i to prove to P_j that they did *not* receive such a message from S. Polarizers must satisfy two important properties of signatures: they are transferable and cannot be forged by an adversary to incriminate honest parties. Unlike a digital signature, however, they can only be generated through an interactive protocol involving the sender S and all possible future witnesses, and the transferability of a polarizer is a context-dependent property that only holds within a protocol execution. The formal definition is provided below.

Definition 9 (Σ-Polarizer). *A Σ-**polarizer** is a 4-tuple*

$$\mathsf{Pol} = (\mathcal{A}, \mathcal{C}, \mathsf{Acc}, \mathsf{MakeGraph}) \tag{15}$$

where

- *$\mathcal{A} \subseteq \mathcal{P}$ is called the set of alive parties;*
- *$\mathcal{C} \subseteq \mathcal{P}$ is called the set of corrupt parties;*
- *Acc is a set of Σ-*accusations *among parties in $\mathcal{P}$;*
- *$\mathsf{MakeGraph}$ is an efficient algorithm that on input a set of Σ-*accusations *among parties in $\mathcal{P}$ outputs an undirected graph $G = (\mathcal{P}, E)$.*

*We say that Pol is **valid** if the following two properties hold:*

- (**Completeness**) *The sets $\mathcal{A}$ and $\mathcal{C}$ form a partition of $\mathcal{P}$, that is $\mathcal{A} \cap \mathcal{C} = \varnothing$ and $\mathcal{A} \cup \mathcal{C} = \mathcal{P}$;*
- (**Justifiability**) *If $G \leftarrow \mathsf{MakeGraph}(\mathsf{Acc})$ then for all $P_i \in \mathcal{A}$ and all $P_j \in \mathcal{C}$ there is no path in G between P_i and P_j.*

Remark 4. A simple option for algorithm $\mathsf{MakeGraph}$ is to remove the edge between P_i and P_j in G if and only if $\textsc{Acc}_{ij} \in \mathsf{Acc}$ or $\textsc{Acc}_{ji} \in \mathsf{Acc}$. With this specialization, our definition coincides with that given in [LN24]. However, with this choice of $\mathsf{MakeGraph}$, a large set Acc is needed to guarantee the justifiability property. We show that one can still obtain justifiability with fewer accusations by employing a more involved $\mathsf{MakeGraph}$ algorithm.

The Send Transferable Message Protocol. We are ready to describe our main protocol. In the following, let $h = n - t$ and let $d = 2n/h$. We briefly introduce some notation used directly below. Let G be an undirected graph and let u and v be nodes in G.

1. The *distance between u and v in G*, denoted by $\mathrm{dist}_G(u, v)$, is defined as the length of the shortest path connecting u and v. We write $\mathrm{dist}_G(u, u) = 0$ and if u and v are disconnected in G then we write $\mathrm{dist}_G(u, v) = \infty$.
2. The *neighborhood of u in G* is defined as $N_G(u) = \{v \in G \mid \mathrm{dist}_G(u, v) \leq 1\}$.
3. The *closed ball of radius r centered in u* is defined as $B_G(u, r) = \{v \in G \mid \mathrm{dist}_G(u, v) \leq r\}$.
4. A *clique* is a complete subgraph of G, that is a set of nodes $\{u_1, \ldots, u_\ell\}$ in G such that for all $i, j \in [1, \ell]$ it holds that $u_i \in N_G(u_j)$. The number ℓ is called the *size* of the clique.
5. The *diameter* of G is defined as $\max\{\mathrm{dist}_G(u, v) \mid u, v \text{ are nodes of } G\}$.

When the graph in question is clear from the context, the subscript G is omitted. For clarity of exposition, we describe a local subroutine that each party runs repeatedly during the protocol execution. The subroutine is named MakeGraph and takes as input a set of valid Σ-accusations Acc among parties in $\mathcal{P}$. It is a simplified version of an algorithm used in the context of randomized broadcast protocols from [WXSD20].

Algorithm MakeGraph$(t, \mathcal{P}; \mathsf{Acc})$

Let $K_{\mathcal{P}}$ denote the complete graph on $\mathcal{P}$.

1: Initialize $G \leftarrow K_{\mathcal{P}}$.
2: For each valid Σ-accusation $\textsc{Acc}_{ij} \in \mathsf{Acc}$ remove edge $\{P_i, P_j\}$ from G.
3: Find an edge e of G which is not part of any clique of size h and remove e from G. Repeat until no such edge exists.[a]
4: Return G.

[a] Despite its conceptual simplicity, this step of MakeGraph is inefficient, as the problem of finding a clique of a given size in a graph is NP-complete. We discuss an efficient variant of the algorithm in Section 4.

Lemma 1 (Antimonotonicity of MakeGraph). *For two sets of valid Σ-accusations with $\mathsf{Acc}_1 \supseteq \mathsf{Acc}_2$ one has $\mathsf{MakeGraph}(\mathsf{Acc}_1) \subseteq \mathsf{MakeGraph}(\mathsf{Acc}_2)$.*

Proof. Let $G_1 = \mathsf{MakeGraph}(\mathsf{Acc}_1)$ and $G_2 = \mathsf{MakeGraph}(\mathsf{Acc}_2)$. Assume by contradiction that there exists an edge $e = (P_i, P_j)$ such that $e \in G_1$ but $e \notin G_2$. There are two cases that lead to $e \notin G_2$:

1. Edge e is removed from G_2 in step 2. This means $\textsc{Acc}_{ij} \in \mathsf{Acc}_2$ and because $\mathsf{Acc}_1 \supseteq \mathsf{Acc}_2$ then $\textsc{Acc}_{ij} \in \mathsf{Acc}_1$. However, that means that e is removed from G_1 in step 2 as well, a contradiction.
2. Edge e is not removed from G_2 in step 2 but it is removed in step 3. Since e is not removed from G_1 this means e is an edge in a clique C of size h in G_1. Therefore, for all nodes P_ℓ and P_r in C it holds that $\textsc{Acc}_{\ell r} \notin \mathsf{Acc}_1$ (or the edge $\{P_\ell, P_r\}$ is removed from G_1 in step 2, a contradiction). Because $\mathsf{Acc}_1 \supseteq \mathsf{Acc}_2$ we get $\textsc{Acc}_{\ell r} \notin \mathsf{Acc}_2$ and hence all edges of clique C are in G_2. This means e is part of a clique of size h in G_2, a contradiction.

□

Next, we show that a graph with n nodes in which every edge is part of a clique of size h cannot have diameter larger than $2n/h$, and use this fact to provide an upper bound on the diameter of any graph resulting from MakeGraph.

This is helpful to provide an upper bound for the running time of our send transferable message protocol. Our proof is a simplified version of one given in [WXSD20], where a slightly tighter bound is proven, and due to space constraints, it can be found in Section A.

Lemma 2. *Let G be a graph with n nodes in which every edge is contained in a clique of size h. Then G has diameter at most $d = 2n/h$.*

Lemma 3 (Diameter of $\mathsf{MakeGraph}(\mathsf{Acc})$). *For any set of valid Σ-accusations Acc, the diameter of $\mathsf{MakeGraph}(\mathsf{Acc})$ is at most $d = 2n/h$.*

Proof. By inspection of algorithm $\mathsf{MakeGraph}$, for any set of valid Σ-accusations Acc it holds that any edge in $\mathsf{MakeGraph}(\mathsf{Acc})$ is part of a clique of size h. Lemma 2 then directly implies the claim. □

The following protocol relies on setup in the form of a public key infrastructure $\mathsf{SetUp}_{\mathrm{PKI}}$ for a signature scheme Σ. This means that for each party P_i a public key pk_i is agreed upon by everyone, and if P_i is honest then they know the corresponding secret key sk_i such that $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{Kgn}()$. We do not impose further restrictions, meaning that the public keys of corrupted parties can depend on the public keys of honest parties. Such a setup can be obtained via a public bulletin board, by having each party sample their own keys and publish their public keys. As discussed in Section 1.1, some form of setup is provably necessary for broadcast when $t \geq n/3$.

Protocol $\Pi_{\mathtt{stm}}\,(t, J_{\mathsf{In}}, \Sigma, \mathsf{MakeGraph}, \mathsf{SetUp}_{\mathrm{PKI}})$

Round 0

1: The sender P^*, with input $x = (m, \pi_{\mathsf{In}})$ computes $\sigma = \mathsf{Sgn}_{\mathsf{sk}^*}(m)$ and sends $(\text{IN}, m, (\pi_{\mathsf{In}}, \sigma))$ to all parties.
2: Each party P_i initializes an empty set $\mathsf{Acc}_i^{(0)} \leftarrow \varnothing$.

Round r for $r \geq 1$.

Each party P_i does:
1: Let $\mathsf{Acc}_i^{(r)} \leftarrow \mathsf{Acc}_i^{(r-1)} \cup \{\text{valid } \Sigma\text{-accusations } \text{ACC}_{jk} \text{ received in round } r\}$ and forward all new valid Σ-accusations to all parties.
2: Receive all messages (IN, m', π') and if both
- $\pi' = (\pi_1', \pi_2')$ with $J_{\mathsf{In}}(P_i, m', \pi_1') = \mathtt{true}$,
- $\mathsf{Vfy}_{\mathsf{pk}^*}(m', \pi_2') = \mathtt{true}$,

then forward (IN, m', π') to all parties, output (m', π'), and terminate.
3: Compute $G_i^{(r)} \leftarrow \mathsf{MakeGraph}\left(\mathsf{Acc}_i^{(r)}\right)$.
4: For all P_j such that

$$P_j \in N_{G_i^{(r)}}(P_i) \cap B_{G_i^{(r)}}(P^*, r-1) \tag{16}$$

compute a valid Σ-accusation against P_j and send it to all parties.
5: If

$$\mathrm{dist}_{G_i^{(r)}}(P_i, P^*) = \infty \tag{17}$$

output (NOMSG, π) and terminate, where
- $\pi = \left(\mathcal{A}, \mathcal{C}, \mathsf{Acc}_i^{(r)}\right)$,
- $\mathcal{C} = \left\{P_j \mid \mathrm{dist}_{G_i^{(r)}}(P_i, P_j) = \infty\right\}$,
- $\mathcal{A} = \mathcal{P} \setminus \mathcal{C}$.

In this section, the security of $\Pi_{\mathtt{stm}}$ is proven with respect to the algorithm $\mathsf{MakeGraph}$ described above, so that we never quantify over this specific parameter of $\Pi_{\mathtt{stm}}$. In Section 4 we describe how

the proofs can easily be adapted when MakeGraph is substituted for a polynomial-time counterpart EfficientMakeGraph.

The first lemma shows that honest parties never accuse each other in an execution of $\Pi_{\texttt{stm}}$, and as a consequence, no efficient adversary can produce valid accusations among honest parties.

Lemma 4 (Accusation Soundness). *For all $t < n$, for all t-transferable justifier predicates J_{In} for $\Pi_{\texttt{stm}}$, for all executions of protocol $\Pi_{\texttt{stm}}$ with $f \leq t$ corruptions, for all honest and all adversarial values x (in the sense of* Definition *3), for all honest P_i and P_j, it holds that x is not a valid Σ-accusation $\textsc{Acc}_{ij}$.*

Proof. First, we claim that if P_i and P_j are honest (meaning the adversary never corrupts P_i or P_j throughout the entire execution of the protocol), then at no point in the protocol will P_i produce a valid Σ-accusation against P_j. Suppose, by contradiction, that P_i produces the first valid Σ-accusation against P_j in Round r for some $r \in [1, d]$. This means

$$P_j \in N_{G_i^{(r)}}(P_i) \cap B_{G_i^{(r)}}(P^*, r-1) \tag{18}$$

in step 4 in the protocol. This implies that P_j has not received a valid message $(\textsc{In}, x', \sigma')$ in round $r-1$ (or else P_j would forward such a message to P_i, who would terminate in step 1 of round r, thanks to the t-transferability of J_{In}). Therefore, in round $r-1$ party P_j sends valid Σ-accusations against all parties

$$P_\ell \in N_{G_j^{(r-1)}}(P_j) \cap B_{G_j^{(r-1)}}(P^*, r-2). \tag{19}$$

Let $\widehat{\mathsf{Acc}}_j^{(r-1)}$ denote this specific set of Σ-accusations and consider

$$\widehat{G}_j^{(r-1)} = \mathsf{MakeGraph}\left(\mathsf{Acc}_j^{(r-1)} \cup \widehat{\mathsf{Acc}}_j^{(r-1)}\right). \tag{20}$$

Because all remaining neighbors of P_j in $\widehat{G}_j^{(r-1)}$ are at distance at least $r-1$ from P^*, then it holds that

$$\mathrm{dist}_{\widehat{G}_j^{(r-1)}}(P^*, P_j) \geq r. \tag{21}$$

However, because P_j sends $\widehat{\mathsf{Acc}}_j^{(r-1)}$ to P_i, and all valid Σ-accusations received by honest parties in every round are forwarded to all other honest parties, it holds that

$$\mathsf{Acc}_i^{(r)} \supseteq \mathsf{Acc}_j^{(r-1)} \cup \widehat{\mathsf{Acc}}_j^{(r-1)} \tag{22}$$

From Equation (22), through Lemma 1 we obtain:

$$G_i^{(r)} \subseteq \widehat{G}_j^{(r-1)}. \tag{23}$$

From this, together with Equation (21) we conclude that

$$\mathrm{dist}_{G_i^{(r)}}(P^*, P_j) \geq \mathrm{dist}_{\widehat{G}_j^{(r-1)}}(P^*, P_j) \geq r, \tag{24}$$

which directly contradicts Equation (18). This concludes the proof of the claim, or in other words, shows that honest values in an execution cannot be accusations among honest parties. To show

that adversarial values can also not be accusations among honest parties, notice that the view of the adversary does not include the secret keys of honest parties. The lemma then follows directly from the claim and the fact that an adversary cannot forge valid Σ-signatures on behalf of honest parties.[11] □

The next lemma shows that parties that are honest throughout the protocol execution form a clique in the graph resulting via $\mathsf{MakeGraph}(\mathsf{Acc})$, and this holds for *any* set of valid accusations Acc that an adversary can come up with during the protocol execution.

Lemma 5 (Honest Clique). *For all $t < n$, for all t-transferable justifier predicates J_{ln} for $\Pi_{\mathtt{stm}}$, for all executions of protocol $\Pi_{\mathtt{stm}}$ with $f \leq t$ corruptions, for all (sets of) honest and adversarial values* Acc *(in the sense of* Definition *3), for all honest parties P_i and P_j, it holds that if* Acc *is a set of valid Σ-accusations and $G = \mathsf{MakeGraph}(\mathsf{Acc})$, then*

$$P_j \in N_G(P_i), \tag{25}$$

or, in other words, honest parties form a clique in G.

Proof. By Lemma 4, for all honest parties P_i and P_j there is no valid Σ-accusation $\textsc{Acc}_{ij} \in \mathsf{Acc}$. This means that the edge $\{P_i, P_j\}$ is not removed from G in step 2 of algorithm $\mathsf{MakeGraph}$. Therefore, in step 3 of algorithm $\mathsf{MakeGraph}$ the edge P_i, P_j is part of the honest clique H whose nodes consist of honest parties in this execution of $\Pi_{\mathtt{stm}}$. Because $|H| \geq h$ by definition (recall that $h = n - t$ is the minimum number of honest parties in *any* execution of $\Pi_{\mathtt{stm}}$), then $\{P_i, P_j\}$ is not removed from G is step 3 either, so that $\{P_i, P_j\} \in G$. □

Output Justification. Lemma 4 and Lemma 5 serve as technical tools to show that the following is a valid justifier predicate (meaning transferable, in the sense of Definition 4) for protocol $\Pi_{\mathtt{stm}}$. Define:

$$J_{\mathtt{stm}}(P_i, m, \pi) = \mathtt{true} \tag{26}$$

if and only if one of the two mutually exclusive conditions hold.

$$\begin{aligned}
&1) \quad m = \textsc{NoMsg} \wedge \pi = (\mathcal{A}, \mathcal{C}, \mathsf{Acc}) \\
&\qquad\qquad \wedge (\mathcal{A}, \mathcal{C}, \mathsf{Acc}, \mathsf{MakeGraph}) \text{ is a valid } \Sigma\text{-polarizer} \\
&\qquad\qquad \wedge P^* \in \mathcal{C} \\
&\qquad\qquad \wedge P_i \in \mathcal{A}. \\
&2) \quad m \neq \textsc{NoMsg} \wedge \pi = (\pi_1, \pi_2) \\
&\qquad\qquad \wedge J_{\mathsf{ln}}(P_i, m, \pi_1) = \mathtt{true} \\
&\qquad\qquad \wedge \mathsf{Vfy}_{\mathsf{pk}^*}(m, \pi_2) = \mathtt{true}.
\end{aligned} \tag{27}$$

Lemma 6 (Transferability of $J_{\mathtt{stm}}$). *For all $t < n$, for all t-transferable justifier predicates J_{ln} for $\Pi_{\mathtt{stm}}$, for all executions of protocol $\Pi_{\mathtt{stm}}$ with $f \leq t$ corruptions, it holds that if $J_{\mathtt{stm}}(P_i, m, \pi) = \mathtt{true}$ for some honest $P_i \in \mathcal{P}$, then $J_{\mathtt{stm}}(P_j, m, \pi) = \mathtt{true}$ for all honest $P_j \in \mathcal{P}$.*

[11] It is possible to turn an efficient adversary that produces a valid accusation among honest parties with non-negligible probability into an efficient one that breaks the unforgeability of the underlying signature scheme with non-negligible probability. Our idealized treatment of signatures is discussed in Section 2.2.

Proof. We distinguish two cases.

1. The first case is that $m = \text{NoMsg}$. Here, if $J_{\mathtt{stm}}(P_i, m, \pi) = \mathtt{true}$, then $(\mathcal{A}, \mathcal{C}, \mathsf{Acc}, \mathsf{MakeGraph})$ is a valid Σ-polarizer and $P_i \in \mathcal{A}$ and $P^* \in \mathcal{C}$. Then, by Definition 9 we know that, if $G = \mathsf{MakeGraph}(\mathsf{Acc})$, it holds that $\text{dist}_G(P_i, P^*) = \infty$. By Lemma 5, for all honest P_j it holds thay $P_j \in N_G(P_i)$. If by contradiction $P_j \notin \mathcal{A}$ since by Definition 9 $\mathcal{A} \cup \mathcal{C} = \mathcal{P}$, this means that $P_j \in \mathcal{C}$. However, this means $\text{dist}_G(P_i, P_j) = 1 < \infty$, which is a contradiction because $P_i \in \mathcal{A}$.
2. The second case is $m \neq \text{NoMsg}$. Here, $J_{\mathsf{ln}}(P_i, m, \pi_1) = \mathtt{true}$ as well as $\mathsf{Vfy}_{\mathsf{pk}^*}(m, \pi_2) = \mathtt{true}$. By transferability of J_{ln} (Definition 4) we get that $J_{\mathsf{ln}}(P_j, m, \pi_1) = \mathtt{true}$ for all honest P_j.

□

We can now prove that for all $t < n$, the triple $(\Pi_{\mathtt{stm}}, J_{\mathsf{ln}}, J_{\mathtt{stm}})$ is a t-secure send transferable message protocol for all t-transferable justifier predicates J_{ln} for $\Pi_{\mathtt{stm}}$. We split the proof into three lemmas, one for each property (justified outputs, justified message, and validity).

Lemma 7 (Justified Outputs of $\Pi_{\mathtt{stm}}$). *For all $t < n$, for all t-transferable justifier predicates J_{ln} for $\Pi_{\mathtt{stm}}$, for all executions of protocol $\Pi_{\mathtt{stm}}$ with $f \leq t$ corruptions, for all honest parties P_i, the output of P_i is of the form (x, π) and it holds that $J_{\mathtt{stm}}(P_i, x, \pi) = \mathtt{true}$.*

Proof. By inspection of the protocol, the output of honest parties (assuming this honest party produces output) in any execution is an ordered couple (x, π). We distinguish three cases:

1. If an honest party P_i produces output (x, π) in step 2 of any round $r \in [1, d+1]$, then this means that $\pi = (\pi_1, \pi_2)$ and that $J_{\mathsf{ln}}(P_i, x, \pi_1) = \mathtt{true}$ and $\mathsf{Vfy}_{\mathsf{pk}^*}(x, \pi_2) = \mathtt{true}$, which means that $J_{\mathtt{stm}}(P_i, x, \pi) = \mathtt{true}$.
2. If an honest party P_i produces output in round $r < d+1$ in step 5, by inspection of the protocol this output is of the form (NoMsg, π), where if the output is produced in round r we have that

$$\begin{aligned} \pi &= \left(\mathcal{A}, \mathcal{C}, \mathsf{Acc}_i^{(r)}\right), \\ \mathcal{C} &= \left\{ P_j \mid \text{dist}_{G_i^{(r)}}(P_i, P_j) = \infty \right\}, \\ \mathcal{A} &= \mathcal{P} \setminus \mathcal{C}, \end{aligned} \tag{28}$$

and it holds that

$$\text{dist}_{G_i^{(r)}}(P_i, P^*) = \infty. \tag{29}$$

From equations (28) and (29) it immediatly follows that $P^* \in \mathcal{C}$. It is also clear that $P_i \in \mathcal{A}$ because by definition $\text{dist}_{G_i^{(r)}}(P_i, P_i) = 0 \neq \infty$. It only remains to prove that

$$\left(\mathcal{A}, \mathcal{C}, \mathsf{Acc}_i^{(r)}, \mathsf{MakeGraph}\right) \tag{30}$$

is a valid Σ-polarizer according to Definition 9. The completeness property follows directly from Equation (28). To prove the justifiability property, let $P_k \in \mathcal{A}$ and let $P_\ell \in \mathcal{C}$. Since $P_k \in \mathcal{A}$, then

$$\text{dist}_{G_i^{(r)}}(P_i, P_k) = d_{ik} < \infty. \tag{31}$$

Suppose by contradiction that

$$\text{dist}_{G_i^{(r)}}(P_k, P_\ell) = d_{k\ell} < \infty. \tag{32}$$

Then it must hold that

$$\mathrm{dist}(P_i, P_\ell) \leq d_{ik} + d_{k\ell} < \infty, \tag{33}$$

which contradicts $P_\ell \in \mathcal{C}$.

□

Lemma 8 (Justified Message of $\Pi_{\mathtt{stm}}$). *For all $t < n$, for all t-transferable justifier predicates J_{In} for $\Pi_{\mathtt{stm}}$, for all executions of protocol $\Pi_{\mathtt{stm}}$ with $f \leq t$ corruptions, for all adversarial justified outputs $\mathbf{y} = (y_1, \ldots, y_n)$ with respect to $J_{\mathtt{stm}}$, it holds that*

- *$y_i = \perp$, or*
- *$y_i = \text{NoMsg}$, or*
- *$y_i = (m_i, \pi_i)$ with $\pi_i = \left(\pi_i^{\mathsf{In}}, \pi_i^{\mathsf{Out}}\right)$ and $J_{\mathsf{In}}\left(m_i, \pi_i^{\mathsf{In}}\right) = \mathtt{true}$.*

Proof. Suppose that $y_i \neq \perp$ and $y_i \neq \text{NoMsg}$. Then, by Definition 5 it holds that $y_i = (m, \pi)$ and $J_{\mathtt{stm}}(m, \pi) = \mathtt{true}$. The lemma follows directly from the definition of $J_{\mathtt{stm}}$ (Equation (27)). □

Lemma 9 (Validity of of $\Pi_{\mathtt{stm}}$). *For all $t < n$, for all t-transferable justifier predicates J_{In} for $\Pi_{\mathtt{stm}}$, for all executions of protocol $\Pi_{\mathtt{stm}}$ with input (m, π) and $f \leq t$ corruptions, for all adversarial justified outputs $\mathbf{y} = (y_1, \ldots, y_n)$ with respect to $J_{\mathtt{stm}}$, it holds that if P^* is honest then $y_i = \perp$ or $y_i = (m, \pi_i)$.*

Proof. Suppose that $y_i \neq \perp$. If P_i is honest, since P^* is honest, P_i receives a valid message $(\text{In}, (m, \pi), \sigma')$ in round 1 and output $y_i = (m, \pi_i)$ with $\pi_i = (\pi, \sigma')$. If P_i is corrupted, suppose that $y_i = (m_i, \pi_i)$. By the definition of adversarial justified output (Definition 5) it holds $J_{\mathtt{stm}}(m_i, \pi_i) = \mathtt{true}$. We distinguish two cases:

1. The first case is $m_i = \text{NoMsg}$. Then by Equation (27) it holds that

$$\pi_i = (\mathcal{A}, \mathcal{C}, \mathsf{Acc}) \tag{34}$$

   such that $P_j \in \mathcal{A}$ for all honest parties P_j and $P^* \in \mathcal{C}$. However, because P^* is honest this contradicts Lemma 5.
2. The second case is $m_i \neq \text{NoMsg}$. In this case, by Equation (27) we have $\pi_i = (\pi_i, \sigma)$ with $\mathsf{Vfy}_{\mathsf{pk}^*}(m_i, \sigma) = \mathtt{true}$. However, by inspection of the protocol, P^* only signs message m during $\Pi_{\mathtt{stm}}$. Since $\mathcal{A}$ cannot forge signatures on $m_i \neq m$ on behalf of an honest P^*, it follows that $m_i = m$.

□

We show that in any execution of $\Pi_{\mathtt{stm}}$ where f is the *actual* number of corruptions, all honest parties terminate within $\min\{f+2, d+2\}$ rounds. If $t < (1-\varepsilon)n$ for any positive constant $\varepsilon \geq 1$, we get $h = n - t \geq \varepsilon \cdot n$ and $d \leq 2/\varepsilon$, so that the protocol terminates in a constant number of rounds.

Lemma 10 (Round Complexity of $\Pi_{\mathtt{stm}}$). *For all $t < n$, all t-transferable justifier predicates J_{In} for $\Pi_{\mathtt{stm}}$, for all executions of protocol $\Pi_{\mathtt{stm}}$ with $f \leq t$ corruptions, we have $r(\Pi_{\mathtt{stm}}) \leq \min\{f+2, d+2\}$. In addition, honest parties terminate at most 1 round apart, that is, for all honest P_i and P_j it holds that $|r(P_i) - r(P_j)| \leq 1$.*

Proof. Consider an execution of $\Pi_{\mathtt{stm}}$ with $f \leq t$ corruptions and suppose that no honest party P_i has terminated in round $f+1$. Because in round f all honest parties P_i send valid Σ-accusations against all parties

$$P_\ell \in N_{G_i^{(f)}}(P_i) \cap B_{G_i^{(f)}}(P^*, f-1), \tag{35}$$

this means that, for all honest P_j

$$\mathrm{dist}_{G_i^{(f+1)}}(P_j, P^*) \geq f+1. \tag{36}$$

Suppose that

$$\mathrm{dist}_{G_i^{(f+1)}}(P_i, P^*) < \infty. \tag{37}$$

Then, there is a path of length at least $f+1$ (with $f+2$ nodes) connecting P_i and P^*. Since the number of corrupted parties is f, the path travels through an honest node $P_j \neq P_i$, so that

$$\mathrm{dist}_{G_i^{(f+1)}}(P_j, P^*) \leq f, \tag{38}$$

which contradicts Equation (36). Now, suppose that some honest party P_i terminates in round $r < d+1$, and consider the following two cases:

1. If P_i terminates upon receiving a valid $(\textsc{In}, (m, \pi), \sigma')$, the message is forwarded to all parties, and by t-transferability of J_{In}, every other honest party terminates in round $r+1$ at the latest.
2. If P_i terminates because

$$\mathrm{dist}_{G_i^{(r)}}(P_i, P^*) = \infty, \tag{39}$$

because all valid Σ-accusations are forwarded by honest parties in all rounds, we have $\mathsf{Acc}_j^{(r+1)} \supseteq \mathsf{Acc}_i^{(r)}$ for all honest P_j. From Lemma 1 it follows that $G_j^{(r+1)} \subseteq G_i^{(r)}$ which in turn implies

$$\mathrm{dist}_{G_j^{(r+1)}}(P_i, P^*) \geq \mathrm{dist}_{G_i^{(r)}}(P_i, P^*) = \infty. \tag{40}$$

If, by contradiction, parties P_j and P^* are connected in $G_j^{(r+1)}$, then by Lemma 5 parties P_i and P^* are connected in $G_j^{(r+1)}$, which contradicts Equation (40).

We have shown that if an honest P_i terminates in round r, then all honest P_j terminate in round $r+1$ at the latest, and that some honest party terminates by round $f+1$ at the latest. To conclude the proof, we show that some honest party P_i terminates in round $d+1$ at the latest. By inspection of algorithm $\mathsf{MakeGraph}$, for each $r \in [1, d+1]$ it holds that each node in $G_i(r)$ is part of a clique of size h. Lemma 3 then guarantees that the diameter of $G_i^{(r)}$ is at most d. Suppose, by contradiction, that no honest party has terminated in round $d+1$. Then, arguing as above, we get

$$\mathrm{dist}_{G_j^{(d+1)}}(P_j, P^*) \geq d+1, \tag{41}$$

a contradiction. □

Corollary 1 (t-Security of $\Pi_{\mathtt{stm}}$). *For all $t < n$ and for all t-transferable justifier predicates J_{In} for $\Pi_{\mathtt{stm}}$, the triple*

$$(\Pi_{\mathtt{stm}}, J_{\mathsf{In}}, J_{\mathtt{stm}})$$

is a t-secure send transferable message protocol.

Proof. Lemma 6 shows that $J_{\mathtt{stm}}$ is a t-transferable justifier predicate for $\Pi_{\mathtt{stm}}$ for all $t < n$. Lemma 7, Lemma 8, and Lemma 9 show that justified outputs, justified message, and validity respectively hold for all $t < n$.

3.2 From Send Transferable Message to Broadcast

Following the construction in [LN24], our protocol $\Pi_{\texttt{stm}}$ directly yields a n-party broadcast protocol that is optimally resilient (can tolerate t corruptions for all $t < n$) and asymptotically matches the lower bound of $\min\{f+2, t+1\}$) rounds for any constant fraction of honest parties, that is if $t < (1-\varepsilon)n$ for some $\varepsilon > 0$.

Lemma 11 (Broadcast from Send Transferable Message). *Suppose that there exists a t-secure send transferable message protocol $\Pi_{\texttt{stm}}$ with $r(\Pi_{\texttt{stm}}) = \ell$. Then there exists a t-secure broadcast protocol $\Pi_{\texttt{bc}}$ with $r(\Pi_{\texttt{bc}}) = O(\min\{\ell \cdot f, t\})$, where $f \leq t$ is the actual number of corruptions in the execution of the protocol.*

Proof. Sections 6 and 7 of [LN24]. □

Theorem 1. *For all $t < n$, there exists a t-secure broadcast protocol $\Pi_{\texttt{bc}}$ such that*

- $r(\Pi_{\texttt{bc}}) = O(\min\{f^2, t\})$;
- $r(\Pi_{\texttt{bc}}) = O(f)$ *if* $t < (1-\varepsilon)n$ *for some* $\varepsilon > 0$.

Proof. Corollary 1 shows that $\Pi_{\texttt{stm}}$ is a t-secure send transferable message protocol for all $t < n$. Lemma 10 shows that $r(\Pi_{\texttt{stm}}) \leq \min\{f+2, d+2\}$. Lemma 11 then implies the existence, for all $t < n$, of a of a t-secure broadcast protocol $\Pi_{\texttt{bc}}$ with

$$r(\Pi_{\texttt{bc}}) = O\big(\min\{f \cdot \min\{f+2, d+2\}, t\}\big) = O(\min\{f^2, t\}). \tag{42}$$

If $t < (1-\varepsilon)n$ for some constant $\varepsilon > 0$, then

$$d = \frac{2n}{h} = \frac{2n}{n-t} = \frac{2n}{\varepsilon n} = \frac{2}{\varepsilon}, \tag{43}$$

which yields

$$r(\Pi_{\texttt{bc}}) = O\left(\min\left\{f \cdot \min\left\{f+2, \frac{2}{\varepsilon}\right\}, t\right\}\right) = O(\min\{f, t\}) = O(f). \tag{44}$$

□

4 Efficient Graph Building Algorithm

In this section, we present algorithm EfficientMakeGraph, a polynomial time analogous of MakeGraph, and we show that when the latter is replaced by the former in $\Pi_{\texttt{stm}}$ the security of the protocol is preserved (Corollary 1). We do not claim the ideas of this algorithm, which we got from [WXSD20], but in general follows from elementary graph-theoretic observations. Recall that for all $t < n$ we define $h = n - t$ and $d = 2n/h$.

Algorithm EfficientMakeGraph$(t, \mathcal{P}; \mathsf{Acc})$

Let $K_{\mathcal{P}}$ denote the complete graph on $\mathcal{P}$.

1: Initialize $G \leftarrow K_{\mathcal{P}}$.
2: For each valid Σ-accusation $\textsc{Acc}_{ij} \in \mathsf{Acc}$ remove edge $\{P_i, P_j\}$ from G.

3: Find an edge $e = \{P_i, P_j\}$ of G such that $|N_G(P_i) \cap N_G(P_j)| < h$ and remove e from G. Repeat until no such edge exists.
4: Return G.

Lemma 12 (Antimonotonicity of EfficientMakeGraph). *For two sets of valid Σ-accusations with $\mathsf{Acc}_1 \supseteq \mathsf{Acc}_2$ one has $\mathsf{MakeGraph}(\mathsf{Acc}_1) \subseteq \mathsf{MakeGraph}(\mathsf{Acc}_2)$.*

Proof. Let $G_1 = \mathsf{MakeGraph}(\mathsf{Acc}_1)$ and $G_2 = \mathsf{MakeGraph}(\mathsf{Acc}_2)$. Assume by contradiction that there exists an edge $e = (P_i, P_j)$ such that e is an edge in G_1 but e is not an edge in G_2. There are two cases:

1. Edge e is removed from G_2 in step 2. This means $\textsc{Acc}_{ij} \in \mathsf{Acc}_2$ and because $\mathsf{Acc}_1 \supseteq \mathsf{Acc}_2$ then $\textsc{Acc}_{ij} \in \mathsf{Acc}_1$. However, that means that e is removed from G_1 in step 2 as well, a contradiction.
2. Edge e is not removed from G_2 in step 2 but it is removed in step 3. Since e is an edge in G_1 this means
$$|N_{G_1}(P_i) \cap N_{G_1}(P_j)| \geq h. \tag{45}$$
For all nodes P_ℓ in $N_G(P_i) \cap N_G(P_j)$ it holds that $\textsc{Acc}_{i\ell} \notin \mathsf{Acc}_1$ and $\textsc{Acc}_{\ell j} \notin \mathsf{Acc}_1$ (or the edges $\{P_i, P_\ell\}$ and $\{P_\ell, P_j\}$ are removed from G_1 in step 2, a contradiction). Because $\mathsf{Acc}_1 \supseteq \mathsf{Acc}_2$ we get $\textsc{Acc}_{i\ell} \notin \mathsf{Acc}_2$ and $\textsc{Acc}_{\ell j} \notin \mathsf{Acc}_2$ and hence
$$P_\ell \in N_{G_2}(P_i) \cap N_{G_2}(P_j), \tag{46}$$
from which we can conclude that
$$|N_{G_2}(P_i) \cap N_{G_2}(P_j)| \geq |N_{G_1}(P_i) \cap N_{G_1}(P_j)| \geq h. \tag{47}$$
Therefore, edge e is not removed from G_2 is step 3, a contradiction.

□

Unfortunately, we cannot directly use Lemma 2 to bound the diameter of the output of EfficientMakeGraph (the equivalent of Lemma 3 for EfficientGraph), because in general it is not guaranteed that every edge in the output of EfficientMakeGraph is part of a clique of size h, as the graph in Figure 1 below shows.

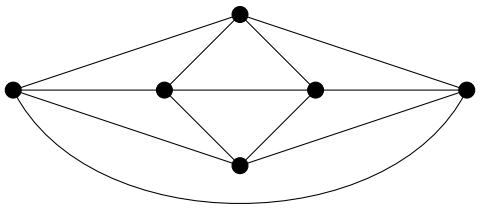


Fig. 1. A graph with 6 nodes where every edge $\{u, v\}$ satisfies $|N_G(u) \cap N_G(v)| \geq 4$ but the curved edge is *not* part of any clique of size 4.

Below we show that the equivalent of Lemma 3 is true for EfficientMakeGraph as well, even though the proof is more involved. Due to space constraints, the proof is presented in B.

Lemma 13 (Diameter of $\mathsf{EfficientMakeGraph}(\mathsf{Acc})$). *For any set of valid Σ-accusations Acc, the diameter of $\mathsf{EfficientMakeGraph}(\mathsf{Acc})$ is at most $d = 2n/h$.*

If we replace algorithm MakeGraph with EfficientMakeGraph in protocol $\Pi_{\mathtt{stm}}$, Lemma 4 still holds, because its proof only relies on the antimonotonicity of MakeGraph (Lemma 1), and EfficientMakeGraph still satisfies this property (as we show in Lemma 12). However, the same is not obvious for Lemma 5, as its proof relies on the specific structure of MakeGraph in addition to Lemma 4. Luckily, it is not hard to show that an analogous of Lemma 5 still holds when we replace algorithm MakeGraph with EfficientMakeGraph in protocol $\Pi_{\mathtt{stm}}$. We do so below.

Lemma 14. *Consider step 3 of algorithm* EfficientMakeGraph *as an algorithm that takes as input a graph G with node set $\mathcal{P}$ and outputs a subgraph G' of G. Then, for all edges e of G which are part of a clique of size h, it holds that e is an edge of G'.*

Proof. Suppose that $e = \{P_i, P_j\}$ is part of a clique C of size h in G. Then, for all nodes P_ℓ of C, it holds that $P_\ell \in N_G(P_i)$ and $P_\ell \in N_G(P_j)$, which in turn means $|N_G(P_i) \cap N_G(P_j)| \geq h$, so that e is not removed from G and e is an edge of G'. □

Lemma 15 (Honest Clique in $\Pi_{\mathtt{stm}}$ with EfficientMakeGraph). *For all $t < n$, for all t-transferable justifier predicates J_{In} for $\Pi_{\mathtt{stm}}$, for all executions of protocol $\Pi_{\mathtt{stm}}$ with $f \leq t$ corruptions, for all (sets of) honest and adversarial values Acc (in the sense of* Definition *3), for all honest parties P_i and P_j, it holds that if Acc is a set of valid Σ-accusations and $G = \mathsf{EfficientMakeGraph}(\mathsf{Acc})$, then*

$$P_j \in N_G(P_i), \tag{48}$$

or, in other words, honest parties form a clique in G.

Proof. By Lemma 4, for all honest parties P_i and P_j there is no valid Σ-accusation $\textsc{Acc}_{ij} \in \mathsf{Acc}$. This means that the edge $\{P_i, P_j\}$ is not removed from G in step 2 of algorithm MakeGraph. Therefore, in step 3 of algorithm MakeGraph the edge P_i, P_j is part of the honest clique H whose nodes consist of honest parties in this execution of $\Pi_{\mathtt{stm}}$. Because $|H| \geq h$ by definition (recall that $h = n - t$ is the minimum number of honest parties in *any* execution of $\Pi_{\mathtt{stm}}$), then by Lemma 14 the edge $\{P_i, P_j\}$ is not removed from G is step 3 either, so that $\{P_i, P_j\} \in G$. □

Therefore, when algorithm MakeGraph is substituted with its efficient version EfficientMakeGraph, the proof of Corollary 1 follows by replacing every invocation of Lemma 1 with one of Lemma 12, every invocation of Lemma 3 with one of Lemma 13, and every invocation of Lemma 5 with one of Lemma 15.

References

[ACD+19] Ittai Abraham, T.-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In Peter Robinson and Faith Ellen, editors, *38th ACM PODC*, pages 317–326. ACM, July / August 2019.

[AD15] Ittai Abraham and Danny Dolev. Byzantine agreement with optimal early stopping, optimal resilience and polynomial complexity. In *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.

[ALPT22] 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.

[BGP92] 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.

[Coa93] Brian A Coan. Efficient agreement using fault diagnosis. *Distributed computing*, 7:87–98, 1993.

[DRS82] 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.

[DRS90] Danny Dolev, Ruediger Reischuk, and H Raymond Strong. Early stopping in byzantine agreement. *Journal of the ACM (JACM)*, 37(4):720–741, 1990.

[DS83] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

[DY83] Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 29(2):198–208, 1983.

[Ezh87] Paul D Ezhilchelvan. Early stopping algorithms for distributed agreement under fail-stop, omission, and timing fault types. *Computing Laboratory Technical Report Series*, 1987.

[FLZL21] Matthias Fitzi, Chen-Da Liu-Zhang, and Julian Loss. A new way to achieve round-efficient byzantine agreement. pages 355–362. ACM, 2021.

[FM88] Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *20th ACM STOC*, pages 148–161. ACM Press, May 1988.

[GGLZ22] 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, May / June 2022.

[GM98] 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.

[KK06] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 445–462, August 2006.

[LF82] Leslie Lamport and Michael Fischer. Byzantine generals and transaction commit protocols. *Unpublished*, April 1982.

[LN24] Julian Loss and Jesper Buus Nielsen. Early stopping for any number of corruptions. In *EUROCRYPT 2024, Part III*, LNCS, pages 457–488, June 2024.

[LSP19] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. In *Concurrency: the works of leslie lamport*, pages 203–226. 2019.

[Mic17] 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.

[Per85] Kenneth J Perry. Early stopping protocols for fault-tolerant distributed agreement. Technical report, Cornell University, 1985.

[PT84] Kenneth J Perry and Sam Toueg. An authenticated byzantine generals algorithm with early stopping. Technical report, Cornell University, 1984.

[PW92] 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.

[Rei85] Rüdiger Reischuk. A new solution for the byzantine generals problem. *Information and Control*, 64(1-3):23–42, 1985.

[SLM+23] 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, May 2023.

[TPS87] Sam Toueg, Kenneth J Perry, and TK Srikanth. Fast distributed agreement. *SIAM Journal on Computing*, 16(3):445–457, 1987.

[WXDS20] 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, November 2020.

[WXSD20] Jun Wan, Hanshen Xiao, Elaine Shi, and Srinivas Devadas. Expected constant round byzantine broadcast under dishonest majority. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 381–411, November 2020.

Supplementary Material

A Proof of Lemma 2

Let d' be the diameter of G. We show $d' \leq d$. Let v_0 and $v_{d'}$ be nodes of G such that $\text{dist}_G(v_0, v_{d'}) = d'$, and let

$$\begin{aligned} e_1 =& \{v_0, v_1\}, \\ e_2 =& \{v_1, v_2\}, \\ &\vdots \\ e_{d'} =& \{v_{d'-1}, v_{d'}\}, \end{aligned} \tag{49}$$

be a path of length d' between v_0 and $v_{d'}$. By assumption, for all $i \in [1, d']$ edge e_i is part of a clique C_i of size h. Any two cliques C_i and C_j with $j - i \geq 2$ must have disjoint sets of nodes. If this is not the case, and v' is a node in both C_i and C_j, there exists edges $\{v_i, v'\}$ and $\{v', v_{j+1}\}$ in G. This means that the path

$$\begin{aligned} e_1 =& \{v_0, v_1\}, \\ &\vdots \\ e_i =& \{v_{i-1}, v_i\}. \\ & \{v_i, v'\}, \\ & \{v', v_{j+1}\}, \\ e_{j+1} =& \{v_{j+1}, v_{j+2}\}, \\ &\vdots \\ e_{d'} =& \{v_d, v'_d\}, \end{aligned} \tag{50}$$

is a path of length $d' - 1$, a contradiction. Therefore, we can write[12]

$$\begin{aligned} 2n &\geq |C_2 \cup C_4 \cup C_6 \cup \ldots| + |C_1 \cup C_3 \cup C_5 \cup ...| \\ &= |C_2| + |C_4| + |C_6| + ... + |C_1| + |C_3| + |C_5| + ... \\ &= |C_1| + |C_2| + ... + |C_{d'}| \\ &= d' \cdot h. \end{aligned} \tag{51}$$

B Proof of Lemma 13

Let $G = \mathsf{EfficientMakeGraph}(\mathsf{Acc})$. By inspection of the algorithm, it is guaranteed that for any pair of nodes u and v in G it holds that

$$|N_G(u) \cap N_G(v)| \geq h. \tag{52}$$

[12] With a common abuse of notation, we improperly identify the clique C_i with its set of nodes.

Assume, by contradiction, that the diameter of G is $d' > d$. Let v_0 and $v_{d'}$ be nodes of G such that $\mathrm{dist}_G(v_0, v_{d'}) = d'$ and let

$$\begin{aligned} e_1 &= \{v_0, v_1\}, \\ e_2 &= \{v_1, v_2\}, \\ &\vdots \\ e_{d'} &= \{v_{d'-1}, v_{d'}\}, \end{aligned} \tag{53}$$

be a path of length d' connecting v_0 and $v_{d'}$. Consider the set of nodes of G that are at distance exactly i from v_0:

$$D_i = \{w \in G \mid \mathrm{dist}_G(v_0, w) = i\}. \tag{54}$$

By definition of dist_G, for all $i \neq j$ it holds that

$$D_i \cap D_j = \varnothing. \tag{55}$$

It is also clear that for all $i \in [1, d']$ one has $v_i \in D_i$ (so that $D_i \neq \varnothing$) and that

$$N_G(v_i) \subseteq D_{i-1} \cup D_i \cup D_{i+1} \tag{56}$$

for all $i \in [0, d' - 1]$.[13] Now consider the graph $G' = (V', E')$ defined as follows:

$$\begin{aligned} V' &= \bigcup_{i=0}^{d'} D_i, \\ E' &= \left(\bigcup_{i=0}^{d'} (D_i \times D_i) \right) \cup \left(\bigcup_{i=0}^{d'-1} (D_i \times D_{i+1}) \right). \end{aligned} \tag{57}$$

From Equation (55) it follows that $\mathrm{dist}_{G'}(v_0, v_{d'}) = d'$, so that the diameter of G' is at least d'. Also, if e is an edge in G', then from Equation (57) there exists $i \in [0, d' - 1]$ such that $e = \{a, b\}$ with $a \in D_i$ and $b \in D_i \cup D_{i+1}$. This means that each edge in E' is part of the clique whose nodes are $D_i \cup D_{i+1}$, whose size we can lower bound combining Equation (52) with Equation (56) as follows. For all $i \in [0, d' - 1]$:

$$\begin{aligned} h &\leq |N_G(v_i) \cap N_G(v_{i+1})| \\ &\leq |(D_{i-1} \cup D_i \cup D_{i+1}) \ \cap \ (D_i \cup D_{i+1} \cup D_{i+2})| \\ &= |D_i \cup D_{i+1}| \\ &= |D_i| + |D_{i+1}|. \end{aligned} \tag{58}$$

We have shown that very edge of G' is part of clique of size h. Therefore, by Lemma 2, the diameter of G is at most d, which contradicts our assumption that $d' > d$.

[13] We adopt the convention $D_{-1} = \varnothing$ and $D_{d'+1} = \varnothing$.