Early-stopping Consensus with Adaptive Bit Complexity

ERICA BLUM, Reed College
CHRISTOPH LENZEN, Aalto University
JULIAN LOSS, Ruhr University Bochum

Protocols for Byzantine agreement are known to be constrained by relatively strong lower bounds on their optimal resilience, round complexity, and communication complexity. Crucially, though, these lower bounds do not immediately rule out the possibility of protocols that are faster and use less communication when the actual number of faults f is less than the maximum number of faults t that can be sustained. *Early-stopping* protocols terminate in a number of rounds proportional to f (rather than t); likewise, protocols with *adaptive communication* incur asymptotically less communication when f is less than t. We present a randomized, early-stopping Byzantine agreement protocol with adaptive communication complexity that terminates in $O(f+1)$ rounds with bit complexity $O((f+1)n\kappa)$ for a failure probability of $2^{-\kappa}$ in a synchronous network with $t < n/2$ faults, assuming a Public Key Infrastructure (PKI). This is achieved against a strongly adaptive adversary, i.e., the attacker can observe all messages in round r, then choose which parties to corrupt in round r, and *then* remove or alter the round-r messages of corrupted parties.

CCS Concepts: • **Theory of computation → Cryptographic protocols**.

Additional Key Words and Phrases: Byzantine agreement, consensus, early-stopping, adaptive complexity

Authors' addresses: Erica Blum, Reed College, ericablum@reed.edu; Christoph Lenzen, Aalto University, christophlen@ru.is; Julian Loss, Ruhr University Bochum, julianloss@gmail.com.

1 INTRODUCTION

Byzantine agreement (BA) is a fundamental distributed systems problem in which a set of parties must reach agreement on a common output, even when a substantial fraction of parties may be faulty. Byzantine agreement protocols have a reputation for being expensive, in terms of both communication complexity and round complexity. This reputation is not unfounded: even given such luxuries as a synchronous network and public key infrastructure (PKI), there are relatively strong lower bounds on the communication complexity and round complexity as a function of n, the number of parties, and t, the number of faults tolerated. For example, one classic result of Dolev and Strong [14] shows that any deterministic BA protocol must incur at least $t + 1$ rounds of communication, and another classic bound due to Dolev and Reischuk [12] states that any deterministic BA protocol must send $\Omega(t^2)$ messages.

It is not so surprising that we must pay a high price (in communication and round complexity) for fault tolerance in protocol executions where there are many faulty parties; however, it seems reasonable to ask if we might get a discount when the actual number of faulty parties, f, is much smaller than the maximum number of faulty parties, t. The lower bounds mentioned above do not rule out the existence of protocols that achieve better performance when f is smaller than t, and indeed, several fruitful lines of work have explored this gap and uncovered new lower bounds and constructions. In particular, *early stopping* protocols [13, 25] terminate in a number of synchronous rounds that is proportional to the actual number of faults, and protocols with *adaptive communication* [8, 10, 26], similarly, enjoy lower communication complexity when f is less than t.

Early stopping has primarily been studied in the context of deterministic protocols. Deterministic protocols are often desirable when the setting permits, because they tend to be simpler than their randomized counterparts while achieving strong correctness and termination guarantees. However, randomized protocols have the advantage of being able to circumvent classic deterministic lower bounds like those mentioned above; concretely, they can achieve constant expected round complexity [7] and bypass the Dolev-Reischuk lower bound on communication complexity [12].

As is often the case, however, there is fine print: a lower bound of Karlin and Yao [21] shows that no randomized synchronous protocol can guarantee correct termination in $O(k)$ rounds with less than $\Omega(f/(kn))^k$ failure probability, even when $f \in O(k)$. For $f = t = \Theta(n)$, this translates to $\Omega(k)^{-k}$, which has been matched in recent work of Ghinea et al. [20], but it is not known if the parameter range $k < f < t$ allows for higher probability of termination. Regarding communication complexity, with additional assumptions one can circumvent the Dolev-Reischuk lower bound [4, 9], but it still applies when the adversary can remove already sent messages by corrupted parties after corrupting them [1].

This pattern, where better communication complexity, round complexity, and even failure probability are achievable individually when f is much less than t, raises the natural question of whether it is possible to design a randomized, synchronous BA protocol that *simultaneously* achieves optimal fault tolerance, early-stopping, and adaptive communication complexity, while paying as little as possible in terms of failure probability, all using only a PKI (as opposed to heavier cryptographic tools). We answer this question in the affirmative by constructing an efficient BA protocol without the use of heavy cryptographic tools.

1.1 Our Contribution

Our central contribution is an early-stopping protocol (denoted Π_{BA}) that succeeds with probability $1 - 2^{-\Omega(\kappa)}$ within $O(f + t)$ rounds using only $O((f + 1)n\kappa)$ bits.

Theorem 1.1. *With probability* $1 - \mathrm{negl}.(\kappa)$*, binary Byzantine Agreement can be solved within* $O(f + 1)$ *rounds of communication, where honest parties send* $O((f + 1)n\kappa)$ *bits.*

Arguably, we work with a minimal set of assumptions and tools to make this possible:

- Our protocol works against a strongly adaptive adversary, and hence is subject to Dolev-Reischuk type communication lower bounds [1]. However, expressed in terms of f, this necessitates $\Omega(tf) = \Omega(nf)$ messages only.
- In addition, Dolev-Reischuk type lower bounds also require sending $\Omega(nt)$ *signatures* [1], regardless of f. We address this by using *threshold signatures,* which allow us to compress signatures by multiple parties. Thus, we can meet this lower bound using few κ-bit words, where κ is the size of a signature. Threshold signatures can be constructed based on a PKI, without reliance on "heavy" cryptographic tools or strong model assumptions.
- As we permit signatures, the classic bound of $t < n/3$ relaxes to $t < n/2$ [24]. We achieve the optimal $t = \lceil (n-1)/2 \rceil$.
- We achieve an extremely small failure probability of $2^{-\Omega(\kappa)}$ regardless of f, outperforming standard randomized algorithms. This is compatible with the Karlin-Yao bound [21], which vanishes if $\kappa > f$.

To achieve this result, our main technical tool is a mechanism that forces dishonest parties to acknowledge the receipt of a message with a signature share. The main idea is to set up the system such that the secret key of party p_i is, in essence, a threshold $t+1$ signature itself, where each party holds a share. If p_j learns that P_i refuses to follow the protocol, it can reveal its share of p_i's key, and if all honest parties do so, honest parties can sign on behalf of p_i. This serves as a communication-efficient mechanism to overcome dishonest parties' attempts to prevent progress.

Note that the above mechanism requires all honest parties to obtain proof that p_i fails to cooperate. We achieve this by a leader "accusing" p_i, presenting a challenge which should prompt it to provide its signature share, and having each party confront P_i with this challenge. P_i then either provides the share or a proof that the accusing party violated the protocol itself (e.g. in the form of a conflicting signature by that party)—or, if it fails to do so, has its secret key revealed. Crucially, every message in this procedure is sent or received by either the accuser or the accused, resulting in $O(n\kappa)$ bits of communication. If the leader is honest and the accused fails to provide the requested signature share, the leader can combine all shares of its secret key and sign on behalf of the accused. Thus, if only f parties are dishonest, we can guarantee progress at a cost of $O(n(f+1)\kappa)$ bits.

In more concrete terms, our approach resolves the following key hurdle. In order to progress, one must prove that adopting a proposed value does not imperil the validity property of the consensus protocol. By forcing each party to provide a signature share corresponding to its input value (or in case of a dishonest party, *any* value), we can prove to any party with an input value claimed by a minority of the parties that it is safe to adopt another value. We conjecture that this technique might prove useful in other settings as well.

Even with this tool in place, achieving the early-stopping property at small communication cost poses an additional challenge. Randomization comes to the rescue by "hiding" from an attacker which edges will be used to subsample the fraction of parties that claim to be ready for termination until they are used for communication. While the strong adversary can prevent the "positive" confirmation of termination to some party by observing who intends to communicate with them and corrupting just these parties, we complement this with a "negative" acknowledgment, i.e., having parties communicate that they are *not* ready to terminate. Together with spreading the "proof of decision" via an expander to amplify the fraction of parties holding it, we are able to construct an efficient termination mechanism.

1.2 Additional Related Work

We briefly highlight a few other related works not already mentioned. We primarily focus on Byzantine agreement, though other work has considered early-stopping and/or adaptive communication for related problems, including Byzantine broadcast [10, 17]. Unless otherwise noted, all results are for the synchronous setting with $t < n/2$ corruptions (assuming PKI). Throughout, λ denotes a statistical security parameter, and communication complexity is given in words, where a word is a message containing a constant number of signatures or signature shares and whose length does not depend on n. (We assume a signature or signature share fits in $O(\kappa)$ bits, where κ is a cryptographic security parameter.)

There is a long line of work on early-stopping consensus [2, 15, 15–17, 25]. In terms of deterministic protocols, Elsheimy, Loss, and Papamanthou [17] recently showed a deterministic early-stopping protocol with round complexity $f + O(\sqrt{f})$ and polynomial communication. On the randomized side, Elsheimy et al. [15] show an early stopping BA protocol for $t = (1-\alpha)n/2$ corruptions that terminates in $(1+\epsilon)f$ rounds (for parameters $0 < \epsilon < 1$ and $0 < \alpha < 1$) with probability $1 - 2^{\lambda}$.

Another line of recent work has advanced the state of the art for adaptive communication complexity [8, 10, 11, 18, 26]. Most recently, Elsheimy, Tsimos, and Papamanthou [18] show a deterministic (non-early-stopping) BA protocol tolerating $t < (1-\epsilon)n/2$ corruptions with complexity $O(n(f+1))$; Constantinescu et al. [11] and Civit et al. [8] show (non-early-stopping) BA protocols tolerating $t < n/2$ corruptions with adaptive communication of $O(n+tf)$ and $O(n(f+1))$, respectively.

1.3 Paper Organization

We introduce the model and preliminaries in Section 2. Moving briskly on toward the main event (our early stopping BA protocol, in Section 3.2), we give a brief introduction to the central building blocks as black-box definitions in Section 3.1, and turn back to their implementation later, in Section 4.

2 PRELIMINARIES

Throughout, we consider a synchronous, authenticated setting with n parties and up to $t < n/2$ Byzantine corruptions. That is, the algorithm proceeds in synchronous rounds, in which parties perform local computations, send messages, and receive messages, where they can identify the sender of each message they receive. The actual number of corruptions in a particular protocol execution is denoted $f \leq t$. The adversary is *strongly adaptive*. That is, in round r, it first observes all messages honest parties intend to send, then decides which parties to corrupt, and then may replace all messages sent by corrupted parties according to its choice.[1]

We denote by κ a security parameter that serves as both a cryptographic security parameter and statistical security parameter. That is, κ must be sufficiently large such that it is computationally infeasible for the attacker to break the signature scheme. At the same time, conditioned on this being true, our protocol succeeds with probability at least $1 - 2^{-\Omega(\kappa)} =: 1 - \text{negl.}(\kappa)$. We require that $\kappa \gg \log n$, so that $n^{O(1)}$ events that occur with probability $1 - \text{negl.}(\kappa)$ individually also jointly occur with probability $1 - \text{negl.}(\kappa)$.

Each protocol message is assumed to be tagged with identifiers for the subroutine instance and (where appropriate) current phase or iteration, so that the receiving party can process the message correctly. For simplicity, this is suppressed in the protocol descriptions. We assume that parties ignore any malformed messages.

[1] This is often called after-the-fact removal.

Threshold Signature Scheme. We assume that the parties have access to a threshold signature scheme, where setup is performed by a trusted dealer prior to the protocol.

Definition 2.1 ((k,n)-Threshold Signature Scheme). A (k,n)-threshold signature scheme ((or k-out-of-n threshold signature scheme) among a set $\mathcal{P} = \{p_1, \dots, p_n\}$ of n parties consists of algorithms Setup, SignShare, VrfyShare, Combine, and Vrfy with the following syntax and behavior:

- $\mathsf{Setup}(1^\kappa, n)$: generates secret keys sk_i and individual public keys pk_i for each party p_i, and a public key pk according to security parameter κ.
- $\mathsf{SignShare}_{sk_i}(m)$: given message m (and secret key sk_i), outputs a signature share σ_i. We call the output σ_i *a (threshold-k) signature share on m under sk_i*.
- $\mathsf{VrfyShare}_{pk_i}(m, \sigma_i)$: given message m and signature share σ_i (and individual public key pk_i), outputs $b \in \{0,1\}$. We say that a signature share σ_i on message m is *valid* if and only if $\mathsf{VrfyShare}_{pk_i}(m, \sigma_i) = 1$.
- $\mathsf{Combine}_{pk}(m, \{\sigma_i\})$: given message m and a set of valid signature shares $\{\sigma_i\}$ on m (under distinct secret keys) such that $|\{\sigma_i\}| \geq k$, outputs a signature Σ. We call the output Σ a *(threshold-k) signature on m*.
- $\mathsf{Vrfy}_{pk}(m, \Sigma)$: given message m and signature Σ (and public key pk), outputs $b \in \{0,1\}$. We say that a signature Σ on message m is *valid* if and only if $\mathsf{Vrfy}_{pk}(m, \Sigma) = 1$.

Prior to the protocol, the trusted dealer runs $\mathsf{Setup}(1^\kappa, n)$ and distributes to each p_i its secret key sk_i, individual public keys $(pk_1, \dots, p_n)$, and the public key pk.

The natural correctness property for the threshold signature scheme is that if setup was performed honestly, for any message m and any party i, $\mathsf{VrfyShare}_{pk_i}(m, \mathsf{SignShare}_{sk_i}(m)) = 1$, and for any message m and set of k or more valid shares $\{\sigma_i\}$, $\mathsf{Combine}_{pk}(m, \{\sigma_i\}) = 1$. The desired security property is that for any choice of k and n, signatures on m are perfectly unforgeable given any set of signature shares $\{\sigma_i\}$ on m such that $|\{\sigma_i\}| < k$.

For convenience, we write $\langle m \rangle_i^k$ as shorthand for the tuple (m, σ_i), where σ_i is a valid threshold-k signature share on m under p_i's secret key; similarly, we write $\langle m \rangle^k$ as shorthand for the tuple (m, Σ) where Σ is a valid threshold-k signature on m. Note that in our protocols, the reconstruction threshold k for the threshold signature scheme varies depending on the intended use. In all cases, the set of possible signers is the set of all parties, i.e., we need not distinguish which parties can contribute signature shares. We assume signatures and keys are of length κ.

Trusted Setup. We assume that the same trusted dealer that performs the setup for the threshold signature scheme also performs some additional setup. First, it generates, signs, and distributes secret shares of each party's secret key sk_i. As the dealer is trusted, its signature provides a simple verification mechanism for key shares, so the signed shares can be verified. The reconstruction threshold is set to $t+1$, so that any $t+1$ shares of the same key sk_i are sufficient to reconstruct it. The security property is that an attacker given $t' \leq t$ shares of sk_i learns nothing about sk_i. (Looking ahead to our protocols, we must maintain the invariant that secret keys of honest parties are never revealed. Given the fault threshold, this is equivalent to no honest party ever revealing its key, its share of its key, or its share of another honest party's key.) We implicitly assume that parties verify all shares and signatures in each protocol message they receive, and ignore any message that contains an invalid share or signature.

The dealer's final job is to sample sets of neighbors that will determine which parties communicate during certain steps in the protocol. We specify these sets of neighbors using sets of edges $E' \subset E$, where $G = ([n], E)$ is the complete directed graph over the n parties. For each bit $b \in \{0,1\}$ and iteration $k \in [\lfloor \log(n/10) \rfloor + 1]$, the trusted dealer presamples a directed edge set $E_{\mathrm{rnd}}(k,b) \subseteq [n]^2$ such that each edge is present with independent probability $\min\{\kappa/n, 1\}$. (This κ is the same

parameter $\kappa \gg \log(n)$ introduced earlier as the cryptographic security parameter and statistical parameter.) The dealer reveals to party p_i its incoming and outgoing edges for each $E_{\text{rnd}}(k, b)$. We assume this information remains private, so that the adversary only learns whether a party p_i is an endpoint of an edge e upon either corrupting one of the two endpoints or seeing p_i send or receive along e during the corresponding step of the protocol.

Fixed expander graph. In addition to the edge sets sampled by the trusted dealer, the protocol also uses a fixed edge set E_{ex} such that $([n], E_{\text{ex}})$ has constant degree and for any at least $n/10$ parties, all but $n/10$ of the remaining parties are in their neighborhood. E_{ex} does not need to be private, so it can be fixed in advance. (There are suitable vertex expanders for this, see e.g. [6].)

Problem Specification. Throughout this work, we fix the resilience threshold to $t := \lceil (n-1)/2 \rceil$. All algorithms and subroutines are required to terminate at all correct parties. That is, for each party p_i, there is a round r such that p_i outputs a value and stops executing the protocol.[2] A protocol has *round complexity* R if all honest parties terminate within R rounds. It has *communication complexity* B if honest parties send in total at most B bits. We remark that our protocols all use small messages, so that long messages by faulty parties can be dropped without requiring to process them.

In this work, we solve the *Consensus* or *Byzantine Agreement* problem.[3]

Definition 2.2 (Byzantine Agreement (BA)). Let Π be an n-party protocol in which each party p_i is given an input $v_i \in \{0,1\}^\ell$ and outputs a value $o_i \in \{0,1\}^\ell$. Π is a *Byzantine Agreement (BA)* protocol if it guarantees the following properties:

- **Consistency:** There exists a value o such that for every honest party p_i, $o_i = o$.
- **Validity:** If there exists v such that $v_i = v$ for every honest party p_i, then $o_i = v$.

For the special case $\ell = 1$ we refer to the task as *Binary Byzantine Agreement.* We refer to a BA protocol as *silent on input s* if the following property holds.

- **Silence:** If $v_i = s$ for every honest party p_i, then honest parties send no messages.

Note that there can be at most one value s for which a protocol is silent, so we will typically omit the "on input s" when talking about silent consensus protocols. We obtain silent consensus protocols from non-silent ones using a technique of Lenzen and Rybicki [22] for the non-cryptographic setting. For the sake of completeness, we demonstrate how to adapt the technique to the cryptographic setting in Appendix A.

3 EARLY-STOPPING BA FROM SIGNATURE COLLECTION AND SILENT BA

This section presents our main result, an early-stopping Byzantine agreement protocol with adaptive communication complexity. We give a brief overview of the core building blocks (Section 3.1) and then continue directly to the full protocol (Section 3.2), deferring constructions of the building blocks until later sections.

3.1 Overview of Building Blocks

The central building blocks of our full early-stopping BA protocol are Signature Collection for Inputs (Definition 3.1), Signature Collection for Decision (Definition 3.2), and Silent BA (Definition 2.2).

3.1.1 Signature Collection. As the names suggest, the purpose of the signature collection protocols is to collect enough signature shares on some context-specific message to form a threshold signature. Formally:

[2]Note that this is a stronger requirement than merely committing to an output value within a bounded number of rounds.
[3]For notational convenience, we do not explicitly state that randomized algorithm may fail to satisfy properties with some probability p, which we will capture by stating to solve a problem with probability $1 - p$ in theorems.

Definition 3.1 (Signature Collection for Inputs). Given an input $\mathrm{op}_i \in \{0,1\}^\ell$ at each party p_i, $i \in [n]$, *$\hat{f}$-resilient Signature Collection for Inputs with Leader Set $L \subseteq [n]$* has each p_i compute an output s_i that is either a valid threshold-$\lceil 3n/4 \rceil$ signature on some $v \in \{0,1\}^\ell$ or $\perp$, where the following properties hold.

- **No equivocation:** There is at most one $v \in \{0,1\}^\ell$ such that signature $\langle v \rangle^{\lceil 3n/4 \rceil}$ is created.
- **Validity:** If $\langle v \rangle^{\lceil 3n/4 \rceil}$ is created for some $v \in \{0,1\}^\ell$, then $\mathrm{op}_i = v$ for some honest party p_i.
- **Liveness:** If $f \leq \hat{f}$, then all parties output $\langle v \rangle^{\lceil 3n/4 \rceil}$ for some $v \in \{0,1\}^\ell$.

Definition 3.2 (Signature Collection for Decision). *$\hat{f}$-resilient Signature Collection for Decision with Leader Set $L \subseteq [n]$* is defined as follows. Each party p_i, $i \in [n]$, has input $\langle v \rangle^{\lceil 3n/4 \rceil}$ for some $v \in \{0,1\}^\ell$ or $\perp$ and computes an output s_i that is either $\langle v \rangle^n$ for some $v \in \{0,1\}^\ell$ or $\perp$, where the following properties hold.

- **Validity:** If $\langle v \rangle^n$ is created for some $v \in \{0,1\}^\ell$, then each honest party received a threshold-$\lceil 3n/4 \rceil$ signature for (b, commit).
- **Liveness:** If $f \leq \hat{f}$ parties are corrupted, all honest parties receive input $\langle v \rangle^{\lceil 3n/4 \rceil}$ for some $v \in \{0,1\}^\ell$, and $\langle v' \rangle^{\lceil 3n/4 \rceil}$ cannot be generated for $v' \neq v$, then honest parties output $\langle v \rangle^n$.

In the interest of moving on to the main event, we comment briefly on the motivation for these definitions, and defer implementations to Section 4. In our full BA protocol, these subroutines are run sequentially (first signature collection for inputs, then signature collection for decision) with the goal of producing a signed value $\langle v \rangle^n$ for some $v \in \{0,1\}^\ell$. Informally, the validity and no equivocation properties together allow such a message to act as 'proof' that all honest parties are ready to output v. However, notice that both definitions only guarantee liveness if fewer than $\hat{f}$ parties are corrupted. The communication complexity and running time of our implementations are proportional to the size of the leader set, which is itself proportional to the estimated corruption threshold $\hat{f}$. Thus, in the BA protocol, parties initially run the subroutines with a very small leader set. If the initial attempt fails, parties repeat the process with increasingly larger leader sets. Combining these subroutines in just the right way turns out to be quite subtle; our approach utilizes one more building block, introduced next.

3.1.2 Silent Byzantine Agreement. In addition to the subroutines for signature collection, our early-stopping BA protocol also uses a *silent Byzantine agreement* protocol (Def. 2.2). It is relatively straightforward to construct a silent BA protocol by applying a "wrapper" to a standard Byzantine agreement protocol; for completeness, we show how to do so in Appendix A.

It is important to emphasize that our full protocol construction is not circular, even though we use BA as a building block for BA. We assume a standard fixed-round BA, such as the BA of Momose and Ren [23], transform it into a silent BA, and then finally use the silent BA to obtain a BA with bells and whistles (i.e., early-stopping and adaptive communication complexity) beyond those of the original. The silent BA acts a fallback mechanism if the number of faults is large.

3.2 Early-Stopping Byzantine Agreement

We are now ready to present our main construction, an early-stopping Byzantine agreement protocol with adaptive bit complexity. The full protocol, denoted Π_{BA}, is shown in Figure 1. It relies on three subroutines: signature collection for inputs (SCI), signature collection for decision (SCD), and silent BA (SilentBA). The protocol uses these subroutines in a black-box manner, however, concrete implementations are provided as Π_{SCI} (Figure 3), Π_{SCD} (Figure 4), and Π_{shh} (Figure 6), respectively.

3.3 Protocol Overview

In the core protocol loop, the honest parties run signature collection for inputs, followed by signature collection for decision, in hopes of producing "proof" of agreement on a value v in the form of a threshold signature $\langle v \rangle^n$. Initially, the leader set is very small, as is the estimated corruption threshold $\hat{f}$. Because the SCI and SCD subroutines only guarantee liveness if $f \leq \hat{f}$ parties are corrupted, progress is not guaranteed. However, $\hat{f}$ doubles with each iteration, until either $\hat{f}$ exceeds f, or f must be so large that we can achieve the desired communication complexity and round complexity by falling back to the silent BA subroutine.

Given the signature collection subroutines as building blocks, the main remaining technical challenge is to make the protocol early-stopping at low communication cost. While it is straightforward to use the signature collection subroutines to ensure an early *decision* by all parties with a concise proof of $O(\kappa)$ bits, it is challenging to ensure that all parties *receive* it when terminating early. Simply sending the proof to all parties would be sufficient, but this would incur $\Omega(n^2\kappa)$ bits of communication.

$\Pi_{\mathbf{BA}}$: Early-Stopping Byzantine Agreement

Input: Each p_i has an input value $v_i \in \{0,1\}^\ell$.

Main: Each p_i executes the following steps, proceeding in synchronous rounds:

(1) Initialize $\mathrm{op}_i := v_i$ and last $= \infty$. (When p_i terminates, it outputs op_i.)

(2) For $k \in [\lfloor \log(n/10) \rfloor]$, let $\hat{f} := 2^k$, and do:

- (a) Let $L := [2\hat{f} + 1]$.
- (b) Run $\hat{f}$-resilient SCI with Leader Set L and input op_i; call the output σ_k. (σ_k is only valid for the current loop iteration; discard it at the end of the iteration.)
- (c) Run $\hat{f}$-resilient SCD with Leader Set L and input σ_k; call the output (if not $\perp$) Σ. (Once received, Σ remains valid forever and is kept for future iterations.)
- (d) If $\langle v \rangle^{\lceil 3n/4 \rceil}$ for some $v \in \{0,1\}^\ell$ was received in this loop iteration (as σ_k or during Step 2c), set $\mathrm{op}_i = v$.
- (e) If $\Sigma = \langle v \rangle^n$ for some $v \in \{0,1\}^\ell$, send Σ along each incident edge of E_{ex}.
- (f) If holding $\Sigma = \langle v \rangle^n$ for some $v \in \{0,1\}^\ell$, send a termination message along each incident out-edge of $E_{\mathrm{rnd}}(k, 1)$. Otherwise, send a continue message along each incident out-edge of $E_{\mathrm{rnd}}(k, 0)$.
- (g) If p_i received termination messages via at least a 4/5-fraction of its in-edges in $E_{\mathrm{rnd}}(k, 1)$, set last $= \min\{\text{last}, k+1\}$.
- (h) If $k > 1$ and p_i neither received continue messages over at least a 4/5-fraction of its in-edges in $E_{\mathrm{rnd}}(k-1, 0)$ nor has last $\neq \infty$, it queries all parties for $\langle \cdot \rangle^n$ (this is executed only once). Honest parties holding such a signature or obtaining it at any later point respond to the query with it (only once for each requesting party). Once p_i receives $\langle v \rangle^n$ for some $v \in \{0,1\}^\ell$ as a reply to such a query, it sends $\langle v \rangle^n$ to all parties, sets $\mathrm{op}_i = v$, and terminates.
- (i) If last $= k$, terminate.

(3) If last $\neq \infty$ or if p_i holds a signature $\langle v \rangle^n$, it runs SilentBA with input $\perp$, where the silent value $s = \perp$. Else, it runs SilentBA with input op_i. If SilentBA terminates with an output other than $\perp$, set op_i to the output of SilentBA. Then terminate.

Fig. 1. An early-stopping Byzantine agreement protocol for $t < n/2$ adaptive corruptions, using subroutines for signature collection on inputs (SCI), signature collection for decision (SCD), and silent BA (SilentBA).

To avoid this, for each iteration of the protocol, we presample a uniformly random directed graph of in- and out-degree $\Theta(\kappa)$, such that each edge is known only to its endpoints. Honest parties use

this graph to tell out-neighbors whether they are ready to terminate or not. If for a node many in-neighbors (say, a 4/5-fraction) claim that they are ready for termination, this means that with probability $1 - 2^{-\Omega(\kappa)}$, at least, say, a 1/10-fraction of parties is honest and ready for termination. This exploits that an attacker has no information on edges in the presampled graph until these messages are either used for communication or one of their endpoints is corrupted.

Parties that are ready for termination also distribute the decision certificate along their incident edges in a vertex expander. Thus, *any* party receiving many termination messages in an iteration implies that *most* parties, e.g. $9n/10$ many, receive a decision certificate during this iteration. As a consequence, either $f \in \Omega(n)$, or in the next iteration all parties will see termination messages from a 4/5-fraction of their neighbors.

This means that we can use seeing a 4/5-fraction of neighbors claiming that they are ready for termination as trigger to terminate one iteration later, provided that we can cover the case that $f \in \Omega(n)$. We address this by a negative acknowledgement: parties that are not ready for termination will send this to their out-neighbors in another random graph sampled in the same manner. If a party does not receive such messages from most neighbors, this implies that almost certainly a 1/10-fraction of nodes is honest and ready for termination. The only plausible explanation for not seeing many termination messages in the next iteration (which would guarantee termination for all parties one iteration later) is hence that $f \in \Omega(n)$. However, in this case we can afford to query all parties for a decision certificate and send it directly to all parties. This also ensures that termination will be triggered at all honest parties.

As a practical remark, instead of the randomized termination approach, which costs a total of $O((f+1)n\kappa)$ bits of communication, one can simply have each party let *all* other parties know when it is ready to terminate. This results in a deterministic algorithm at the expense of an additive n^2 bits. For a typical signature length of at least 128 bits, this is likely to be dominated by the $O((f+1)n\kappa)$ term of the randomized solution up to $n = 10^5$ or more.

There is one final (and very important) issue related to termination, which we will briefly preview here and then defer fixing until Section 3.6. In a nutshell, the issue is that it is possible for some honest parties to terminate Π_{BA} early while others continue the main loop, but the properties of the signature collection subroutines are not guaranteed to hold if not all honest parties participate. In Section 3.6, we describe a minor modification to the protocol and subroutines that ensures the signature collection subroutines cannot "go rogue" when some honest parties have already terminated. (We opt to present the main protocol without this detail and return to fix it later because (a) it significantly simplifies the presentation of the protocol and subroutines, and (b) by the time we return to it, we will have all the tools we need to fix it.)

3.4 Correctness

We begin by showing correctness. Our first lemma proves that if all honest parties p_i hold the same value $\text{op}_i = v$ at the start of some iteration, then except with negligible probability, this value "sticks" for all future iterations (and no signature $\langle v' \rangle^{\lceil 3n/4 \rceil}$ or $\langle v' \rangle^n$ for $v' \neq v$ will be produced in any future iteration).

Throughout, claims are conditional on subroutines meeting the requirements of Definitions 3.1 and 3.2 and a BA subroutine with the silence property (Definition 2.2).

Lemma 3.3. *If for some $v \in \{0,1\}^\ell$ we have that $\text{op}_i = v$ for each honest party p_i at the beginning of iteration k, with probability $1 - \text{negl.}(\kappa)$ this property becomes an invariant. Moreover, no $\langle v' \rangle^{\lceil 3n/4 \rceil}$ or $\langle v' \rangle^n$ signatures for $v' \neq v$ can be generated in any iteration $k' \geq k$.*

Proof. We show by induction on $k' \geq k$ that (i) no honest party p_i changes op_i in iteration k', (ii) no $\langle v' \rangle^{\lceil 3n/4 \rceil}$ signature for $v' \neq v$ can be generated based on iteration k', and (iii) no $\langle v' \rangle^n$ signature for $v' \neq v$ can be generated based on iteration k'.

For the base case: by assumption, $\text{op}_i = v$ for each honest party p_i at the beginning of iteration k. Hence, the validity property of Definition 3.1 guarantees that no $\langle v' \rangle^{\lceil 3n/4 \rceil}$ signature is generated in Step 2b, showing (ii). Therefore, no honest party changes op_i during iteration k, so (i) holds for k. By the validity property of Definition 3.2, (iii) readily follows for k as well.

For the induction step from $k' \geq k$ to $k' + 1$, observe that the induction hypothesis yields that $\text{op}_i = v$ for each honest party p_i at the beginning of iteration k. Hence, the induction step is completed analogously.

It remains to show that Step 3 does not result in any honest party p_i changing op_i. This is immediate from the validity property of Definition 2.2 if all honest parties participate in the execution of SilentBA. However, we need to cover the case in which at least one honest party has already terminated.

We distinguish two subcases, based on when the honest party terminated. First, consider the subcase where some honest party terminated when executing Step 2h. As part of the same step, it sent a signature $\langle v' \rangle^n$ to all parties. Hence, all participating honest parties p_i use input $\perp$ in SilentBA. By the silence property of Definition 2.2 (and a trivial indistinguishability argument), participating parties cannot distinguish the execution from one in which all honest parties participate with input $\perp$, i.e., no messages are sent and, by validity of Definition 2.2, all parties output $\perp$. Hence, no honest party p_i changes op_i.

The second subcase is that no honest party terminated when executing Step 2h, i.e., some honest party terminated when executing Step 2i. This entails that some honest party received termination messages over at least a 4/5-fraction of their in-edges from $E_{\text{rnd}}(k, 1)$ for some $k < \lfloor \log(n/10) \rfloor$.

Recall that $E_{\text{rnd}}(k, 1)$ is independent from the execution up to Step 2f, i.e., the adversary has no information on which edges are in this set until they are used, except for those incident to corrupted parties. Hence, in expectation, the fraction of in-edges over which a termination message is received is at most $(f + \alpha(n - f))/n$, where α is the fraction of uncorrupted parties that send termination messages in the respective iteration. Assume for contradiction that $(1 - \alpha)(n - f) > n/4$, i.e., more than a quarter of the parties are both honest and send a continuation message. Then

$$\frac{f + \alpha(n - f)}{n} < \frac{f + n - f - n/4}{n} = \frac{3n}{4}.$$

However, by a Chernoff bound and the independence of the probability of $e \in E_{\text{rnd}}(k, 1)$ for each individual edge e, the probability that any honest party receives termination messages over a $4n/5$ fraction of their (in expectation) $\Theta(\kappa)$ incident edges in $E_{\text{rnd}}(k, 1)$ is then $\text{negl.}(\kappa)$. In other words, with probability $1 - \text{negl.}(\kappa)$, $(1 - \alpha)(n - f) > n/4$. In this case, a symmetric argument regarding $E_{\text{rnd}}(k, 0)$ shows that the probability that *any* honest party receives continue messages over a 4/5-fraction of their incident edges in $E_{\text{rnd}}(k, 0)$ is $\text{negl.}(\kappa)$.

In summary, with probability $1 - \text{negl.}(\kappa)$, for each honest party p_i, one of two events occur:

(1) In Step 2g of iteration $k + 1$, p_i receives termination messages via at least a 4/5-fraction of its in-edges in $E_{\text{rnd}}(k + 1, 1)$. Then it either terminates before Step 3 or uses input $\perp$ when running SilentBA.
(2) In Step 2h of iteration $k + 1$, p_i queries all parties for signatures $\langle \cdot \rangle^n$ and, regardless of the corruption choices of the adversary, will receive one. Hence it terminates before Step 3.

Thus, all honest parties p_i that participate in the execution of SilentBA in Step 3 use input $\perp$ and we can reason analogously to the first subcase to complete the proof. □

Corollary 3.4. *Π_{BA} satisfies the validity property of Definition 2.2 with probability* $1 - \mathrm{negl.}(\kappa)$.

Proof. Suppose that for $v \in \{0,1\}^\ell$, we have that $v_i = v$ for each honest party p_i. By Lemma 3.3, with probability $1 - \mathrm{negl.}(\kappa)$ then all honest parties maintain that $\mathrm{op}_i = v$ throughout the algorithm and hence output v. □

Lemma 3.5. *If a $\langle v \rangle^n$ signature is created for some $v \in \{0,1\}^\ell$, then with probability* $1 - \mathrm{negl.}(\kappa)$ *v is unique and $\mathrm{op}_i = v$ becomes an invariant for all honest parties p_i.*

Proof. Suppose that k is the minimum iteration based on which such a signature is created. By the validity property of Definition 3.2, each honest party received a valid $\langle v \rangle^{\lceil 3n/4 \rceil}$ signature during Step 2c of iteration k. To be valid, such a signature must have been created based on Step 2b of iteration k, for which the no equivocation property of Definition 3.1 guarantees that v is unique. Hence, no $\langle v' \rangle^n$ signature can be generated based on iteration k. Moreover, each honest party p_i adopts $\mathrm{op}_i = v$ in Step 2d of iteration k. The claim now follows from Lemma 3.3. □

Lemma 3.6. *Π_{BA} satisfies the agreement property of Definition 2.2 with probability* $1 - \mathrm{negl.}(\kappa)$.

Proof. If a $\langle v \rangle^n$ signature is created for some $v \in \{0,1\}^\ell$, the claim follows from Lemma 3.5.

Otherwise, no honest party sends a termination message. Analogously to the proof of Lemma 3.3, in this case we can show that the probability for any honest party to receive termination messages over at least a 4/5-fraction of its incident edges in $E_{\mathrm{rnd}}(k,1)$ for any k is $\mathrm{negl.}(\kappa)$. Hence, no party ever sets last $\neq \infty$, implying that no party terminates early in Step 2i. Without a $\langle v \rangle^n$ signature, also no party can terminate early due to Step 2h. Hence, all honest parties participate in the BA instance of Step 3. Because there is no $\langle v \rangle^n$ signature and no party has last $\neq \infty$, each honest party p_i inputs op_i and adopts the output value of the BA instance. By the agreement property of BA, this value is identical for all parties. □

3.5 Termination

We now show that Π_{BA} is early-stopping. More precisely, we show that it terminates within $O(f+1)$ rounds, conditional on fast subroutines meeting the requirements of Definitions 3.1 and 3.2 and an $O(n)$-round BA subroutine with the silence property (Definition 2.2). To this end, denote by $T_1(\hat{f}) \in O(\hat{f}+1)$, $T_2(\hat{f}) \in O(\hat{f}+1)$, and $T_3 \in O(n)$ the respective round complexities, for the first two subroutines parametrized by the fault threshold $\hat{f}$ used in the respective iteration.

Lemma 3.7. *If any honest party terminates in iteration $k < \lfloor \log(n/10) \rfloor - 2$ of Step 2 and $f \leq n/10$, then with probability* $1 - \mathrm{negl.}(\kappa)$*, (i) by the end of iteration k, each honest party either terminates, sets last $\leq k+1$, or receives a message of form $\langle v \rangle^n$, and (ii) all honest parties terminate by the end of iteration $k+2$.*

Proof. Let p_i be the first honest party to terminate, in some iteration $k < \lfloor \log(n/10) \rfloor$. If any p_i terminated in Step 2h of iteration k, then all honest parties receive a signature $\langle v \rangle^n$. Analogously to the proof of Lemma 3.3, with probability $1 - \mathrm{negl.}(\kappa)$, no party receives continuation messages over a 4/5-fraction of its incident edges in $E_{\mathrm{rnd}}(k,0)$ in iteration k. Thus, they will either set last $\leq k+1$ before reaching Step 2h in iteration $k+1$, causing termination no later than iteration $k+2$, or terminate in Step 2h of iteration $k+1$, as they already received $\langle v \rangle^n$.

The other case is that p_i terminated in Step 2g. Hence, it received termination messages from a 4/5-fraction of in-neighbors w.r.t. $E_{\mathrm{rnd}}(k-1,1)$. Analogously to the proof of Lemma 3.3, in expectation each party receives termination messages on a fraction of at most

$$\frac{f + \alpha(n-f)}{n} \leq \frac{1}{10} + \frac{9\alpha}{10}$$

of its incident edges in $E_{\text{rnd}}(k-1,1)$, where α is the fraction of honest parties that send termination messages. If $\alpha \leq 1/2$, we can conclude by a Chernoff bound that the probability that p_i terminates is negl.(κ).

Otherwise, at least $(1-\alpha)(n-f) \geq 9n/20$ parties hold a signature $\langle v \rangle^n$ when executing Step 2e in iteration $k-1$. By analogous reasoning, the probability that *any* party receives continuation messages over a 4/5-fraction of its incident edges in $E_{\text{rnd}}(k-1,0)$ in iteration $k-1$ is negl.(κ). Hence, any party either sets last $\leq k+1$ before reaching Step 2h in iteration k or it will query all parties for a signature $\langle v \rangle^n$ in this step, receive one (as no party has terminated yet and honest parties hold signatures $\langle v \rangle^n$), and terminate by the end of iteration k. Either way, all parties terminate by the end of iteration $k+1$ with probability $1-$ negl.(κ). □

Lemma 3.8. *Suppose that for some $k \in [\lfloor \log(n/10) \rfloor - 2]$, we have that $f \leq 2^k$. Then with probability $1-$ negl.(κ), (i) the honest parties receive a signature $\langle v \rangle^n$ in Step 2c of iteration k and (ii) all honest parties terminate by the end of iteration $k+2$.*

Proof. If some honest party terminates by the end of iteration k, the claim follows from Lemma 3.7. Otherwise, all honest parties faithfully execute the k-th iteration, in which $f \leq \hat{f} = 2^k$. Thus, in Steps 2b and 2c, we can apply the liveness properties of Definitions 3.1 and 3.2. This shows that the honest parties obtain a signature $\langle v \rangle^{\lceil 3n/4 \rceil}$ for for some $v \in \{0,1\}^\ell$ in Step 2b and hence a signature $\langle v \rangle^n$ in Step 2c of iteration k, showing (i). Thus, analogously to the previous lemma, the honest parties terminate by the end of iteration $k+2$ with probability $1-$ negl.(κ), showing (ii). □

Corollary 3.9. *If $T_1(\hat{f}), T_2(\hat{f}) \in O(\hat{f}+1)$ and $T_3 \in O(n)$, Π_{BA} terminates within $O(f+1)$ rounds with probability $1-$ negl.(κ).*

Proof. If $f \leq n/80$, then by Lemma 3.8, the algorithm terminates by the end of iteration $k = \max\{1, \lceil \log(f) \rceil\} + 2$. Hence, it runs for $\sum_{k'=1}^{k+2}(T_1(2^{k'}) + T_2(2^{k'})) = O(f+1)$ rounds. Otherwise, $f \in \Omega(n)$ and it runs for $T_3 + \sum_{k'=1}^{\lfloor \log(n/10) \rfloor}(T_1(2^{k'}) + T_2(2^{k'})) = O(n) = O(f+1)$ rounds. □

3.5.1 Bit Complexity. We now show that Π_{BA} is also communication-efficient, provided that $\ell = O(\kappa)$ and the used subroutines are communication-efficient, too. Denote by $C_1(\hat{f}) \in O(n(\hat{f}+1)\kappa)$, $C_2(\hat{f}) \in O(n(\hat{f}+1)\kappa)$, and $C_3 \in O(n^2\kappa)$ the communication complexities of the subroutines solving Definitions 2.2, 3.1 and 3.2 (with the silence property) respectively.

Lemma 3.10. *When executing Π_{BA}, the total number of bits sent by honest parties excluding Step 2h is $O(n(f+1)\kappa)$ with probability $1-$ negl.(κ).*

Proof. By Lemma 3.8, with probability $1-$ negl.(κ) all parties terminate by the end of iteration $k_f \in \log f + O(1)$. The communication cost of the subroutines for Definitions 3.1 and 3.2 is asymptotically dominated by that of iteration k_f, i.e., both require in total $O(n(f+1)\kappa)$ bits. As E_{ex} has constant degree, the total number of bits spent Step 2e is $O(nk_f\kappa) = O(n(f+1)\kappa)$. Similarly, for each iteration $k \leq k_f$ and $b \in \{0,1\}$, we have that $|E_{\text{rnd}}(k,b)| \in \Theta(n\kappa)$ with probability $1-$ negl.(κ) by a Chernoff bound. Thus, the total number of (single-bit) messages using these graphs is $O(nk_f\kappa) = O(n(f+1)\kappa)$. Finally, if any honest party executes Step 3, $k_f \geq \log n - O(1)$, i.e., $f = \Theta(n)$. Thus, the communication complexity of the silent BA routine adds $O(n^2\kappa) = O(n(f+1)\kappa)$. □

For the remaining step, we need to control how many parties the dishonest parties can force them to execute it. This is where the expansion property of E_{ex} is exploited. However, we first show two helper lemmas controlling for how many parties $f \leq \varepsilon n$ nodes can cover a 2ε-fraction of their neighborhood in $E_{\text{rnd}}(k,b)$. These can be viewed as standard results on random graphs, which we include here for completeness.

Lemma 3.11. *Fix a constant $\varepsilon > 0$ and suppose that $f \leq \varepsilon n$. With probability $1 - \text{negl.}(\kappa)$, the total number of honest parties (over all choices of k and b) for which the adversary controls at least a (2ε)-fraction of the in-neighbors of honest parties p_i w.r.t. $E_{\text{rnd}}(k,b)$ by the end of iteration k is $O(f)$.*

Proof. If $\kappa \geq n$, then each edge is "sampled" into each $E_{\text{rnd}}(k,b)$ with probability 1 and the statement holds trivially true. Therefore, assume that $\kappa < n$ in the following.

Fix an iteration k and suppose that the adversary chooses to corrupt f_k parties in this iteration. Prior to this iteration, no information about $E_{\text{rnd}}(k,b)$ for $b \in \{0,1\}$ is revealed by honest parties in any way, and corrupting a party reveals only its incident edges to the adversary. Hence, w.l.o.g. we may assume that the at most f preceding corruptions were performed independently of $E_{\text{rnd}}(k,b)$. By a Chernoff bound, for each party p_i, the probability that at least a $(3\varepsilon/2)$-fraction of in-neighbors of p_i w.r.t. $E_{\text{rnd}}(k,b)$ have been corrupted in preceding iterations is $\text{negl.}(\kappa)$. By a union bound, this holds for all honest parties p_i.

By another Chernoff and union bound, all in- and out-degrees of parties w.r.t. $E_{\text{rnd}}(k,b)$ are in $\Theta(\kappa)$ with probability $1 - \text{negl.}(\kappa)$. Hence, with f_k corruptions, the adversary can increase the fraction of corrupted neighbors of honest parties by $\varepsilon/2$ for no more than $O(f_k/\varepsilon) = O(f_k)$.

Overall, another union bound yields that with probability $1 - \text{negl.}(\kappa)$, the adversary cannot control at least a (2ε)-fraction of the in-neighbors w.r.t. $E_{\text{rnd}}(k,b)$ by the end of iteration k for more than $O(\sum_k f_k) = O(f)$ honest parties p_i, summed over all k and b. □

Lemma 3.12. *Fix a constant $\varepsilon > 0$ and suppose that $f \leq \varepsilon n$. With probability $1 - \text{negl.}(\kappa)$, there is no $k \in [\lfloor \log(n/20) \rfloor]$ and $b \in \{0,1\}$ such that $i \in [n]$ such that for more than εn honest parties, more than a (2ε)-fraction of their in-neighbors w.r.t. $E_{\text{rnd}}(k,b)$ are corrupted.*

Proof. If $\kappa \geq n$, then each edge is "sampled" into each $E_{\text{rnd}}(k,b)$ with probability 1 and the statement holds trivially true. Therefore, assume that $\kappa < n$ in the following.

Fix a subset $F \subset [n]$ of parties of size $|F| = \lfloor \varepsilon n \rfloor$. Let $X_{k,b,i}$ be a random variable indicating whether for fixed k, b, and i we have that party p_i has more than a (2ε)-fraction of its in-neighbors w.r.t. $E_{\text{rnd}}(k,b)$ in F. As edges are chosen uniformly and independently at random into the edge sets, the $X_{k,b,i}$ are independent. Moreover, a Chernoff bound yields that $P[X_{k,b,i} = 1] \leq p \in e^{-\Omega(\kappa)}$. Hence, with $X := \sum_{k,b,i} X_{k,b,i}$, X is upper bounded by a binomially distributed variable with parameters $n' \in O(n \log(n))$ and p. The probability for this variable to attain value $j \geq \varepsilon n$ is upper bounded by

$$\binom{n'}{j} p^j \leq \left(\frac{en'p}{j}\right)^j \leq \left(\frac{en'p}{\varepsilon n}\right)^j = O\left(\frac{\log(n)}{\kappa}\right)^j.$$

As $\kappa \gg \log(n)$, we can bound the $O(\log(n)/\kappa)$ base from above by $2^{-c/\varepsilon}$ for a sufficiently large constant c. It follows that the probability for the variable to exceed εn is at most

$$\sum_{j > \varepsilon n} 2^{-j} \leq 2^{-cn+1}.$$

A union bound over all

$$\binom{n}{\lfloor \varepsilon n \rfloor} \leq \left(\frac{en}{\varepsilon n}\right)^{\varepsilon n} \leq 2^{cn/2},$$

where we use that c is a sufficiently large constant, now concludes the proof. □

Lemma 3.13. *With probability $1 - \text{negl.}(\kappa)$, Π_{BA} has a bit complexity of $O(n(f+1)\kappa)$.*

Proof. By Lemma 3.10, it remains to control the bit complexity of Step 2h. If $f \in \Omega(n)$, this is trivial, as each party sends to each other party at most one signed value $\langle v \rangle^n$ of $O(\kappa + \ell) = O(\kappa)$ bits due to this steps. Otherwise, we may assume that $f \leq n/40$ and apply Lemmas 3.11 and 3.12 with $\varepsilon = 1/50$ to show that for all but $\min\{O(f), n/50\}$ parties, the adversary controls only a

$1/25$-fraction of their incident edges in $E_{\text{rnd}}(k, b)$ for each k and b. Clearly, such parties cannot exhaust our communication budget due to sending queries in Item 2h, and responses to queries from dishonest parties cannot do so either.

Thus, it remains to consider honest parties for which the adversary controls less than a $1/25$-fraction of their in-neighbors w.r.t. $E_{\text{rnd}}(k', b)$ for any $k' \in [k]$ and $b \in \{0, 1\}$. Let k' be the first iteration in which $n/10$ honest parties hold a signature $\langle v \rangle^n$ by Step 2f. As $1/10 + 1/20 + 1/50 < 1/5$, by a Chernoff bound with probability $1 - \text{negl.}(\kappa)$ each of the considered honest parties receives continue messages from at least a $4/5$-fraction of their in-neighbors from $E_{\text{rnd}}(k'', 0)$ for $k'' < k$. On the other hand, due to Step 2e and the vertex expansion of E_{ex}, all but $n/10$ honest parties hold a decision certificate by Step 2e of iteration $k' + 1$. As all but $n/50$ honest parties have not terminated yet and $f \leq n/50$, using that $1/10 + 1/20 + 2/50 < 1/5$, another Chernoff bound shows that with probability $1 - \text{negl.}(\kappa)$ the considered parties receive termination messages from a $4/5$-fraction of their in-neighbors w.r.t. $E_{\text{rnd}}(k' + 1, 1)$. Therefore, they never send queries in Step 2h. □

3.6 Security and Efficiency of Building Blocks under Early Termination

Earlier, we foreshadowed that a subtle issue arises when plugging our signature collection subroutines into Π_{BA}, namely that parties may terminate Π_{BA} early, leaving some honest parties still running the various subroutines.[4] To see why this is problematic, suppose an honest party p_j has already terminated BA, and other honest parties are now running signature collection for inputs in the next iteration. Our proof of security for leader challenge crucially relies on the fact that an honest party is online to defend itself by providing a valid response. As-is, p_j could be challenged (even by another honest party!), leading other honest parties to reveal their share of sk_j. If it is possible for even a single honest party to reveal its secret share of another honest party's key, then things might go very badly.

Fortunately, if an honest party has terminated early, then the signature collection subroutines have already served their purpose by producing a signature $\langle v \rangle^n$ on some value v. In particular, Lemma 3.7 show that if any party terminates during an iteration, then before the next call to signature collection, each party either terminates or is set to do so within two iterations (without relying on liveness of the signature collection subroutines).[5] This means it is perfectly fine for the signature collection subroutines to lose liveness, as long as we maintain the following properties: (i) the communication complexity and round complexity bounds of the subroutines still hold, and (ii) no honest party reveals its secret share of another honest party's key. The former ensures that Π_{BA} retains its desired efficiency, and the latter ensures that the signature collection subroutines cannot produce any invalid output to "compete" with the value v carried by the threshold-n signature, so that when parties terminate, it cannot be with any value besides v.

To achieve this, we make the following relatively simple modification to the protocol: once a party has set $\text{last} < \infty$ or received a signature of form $\langle v \rangle^n$, it will no longer reveal its secret share of any honest party's secret key during the leader challenge protocol. If $\text{last} < \infty$, this is also proof that termination is already underway, so that it is safe to simply refuse to take part in the leader challenge protocol altogether. On the other hand, if a party has received a signature $\langle v \rangle^n$, it has something better to show for guaranteeing progress: rather than forwarding responses or revealing key shares or sending other protocol messages (excepting termination messages, which must be kept short), it can simply present the recipient with the signature $\langle v \rangle^n$. This means that the signature propagates rather than any less valuable information, maintaining liveness in this case.

[4] This issue does not arise for silent BA, because if any honest party has terminated, then all honest parties will run silent BA with input $s = \perp$, and in such an execution the silence property guarantees that a terminated party behaves identically to a still-running honest party.

[5] Formally, this is not given by the statement of the lemma, but it readily follows from its proof.

To see why this modification gives us the properties we need, first notice that the round complexity of our (fixed-round) subroutines does not change, and a 1-to-1 replacement of each length-$O(\kappa+\ell)$ message with another length-$O(\kappa+\ell)$ message has no effect on communication complexity, so property (i) holds. For (ii), recall that by Lemma 3.7, if any honest party terminates in an iteration k of the main loop, then each other party either terminates, sets last $< \infty$, or receives a signature of form $\langle v \rangle^n$ before it would next run signature collection. This is exactly what was needed for (ii): we've ensured that as soon as some honest party terminates, any remaining honest parties stop revealing secret shares. The end result is that the signature collection protocols can no longer cause any harm even if some honest parties have terminated, and yet the original termination guarantees remain intact.

3.7 Putting Everything Together

Our main result is now obtained by using the subroutines we develop in Section 4 and the silent BA routine we give in Appendix A in protocol Π_{BA}, after applying the modifications discussed in Section 3.6.

Theorem 1.1. *With probability* $1 - \text{negl.}(\kappa)$*, binary Byzantine Agreement can be solved within* $O(f+1)$ *rounds of communication, where honest parties send* $O((f+1)n\kappa)$ *bits.*

Proof. By Theorems 4.10, 4.15 and A.3 (where $\ell = 1$ for binary BA), we have subroutines of suitable complexities to plug into Π_{BA}, making the modifications described in Section 3.6. Corollary 3.4 and Lemma 3.6 then show correctness, Corollary 3.9 the running time bound, and Lemma 3.13 the communication complexity bound. □

4 IMPLEMENTING SIGNATURE COLLECTION

We now turn back to the task of implementing the signature collection subroutines used in the early-stopping BA protocol. The central difficulty lies in the fact that (i) for inputs, we need all parties to provide signature shares on their input in order to guarantee that some coordinating party can prove that there is a value that can be safely adopted without violating validity and (ii) for a decision, we want signature shares from all parties as proof that they have observed (and hence adopted) the decision value. We overcome this difficulty using a *leader challenge* subroutine. It provides us with a tool to force f uncooperative parties to either provide their shares or have their secret keys revealed, at a communication cost proportional to f.

In the rest of this section, we implement the leader challenge subroutine (Section 4.1), and then use it to construct Signature Collection for Inputs (Section 4.2) and Signature Collection for Decision (Section 4.3) in quick succession.

4.1 Leader Challenge Subroutine

We now introduce the *Leader Challenge* subroutine, which will form the backbone of our signature collection protocols. The leader challenge subroutine encapsulates the process of challenging a party who did not provide a valid signature share during a previous step. Informally, the purpose of this subroutine is to ensure all honest parties receive a threshold signature share under the private key of the party being challenged. This share can be created in one of two ways: either the party being challenged cooperates by responding with a valid signature share, or the honest parties reveal their shares of the challenged party's secret key, allowing an honest party to create the share on behalf of the challenged party. Critically, our protocol ensures that honest parties never reveal a share of an honest party's secret key—an honest party will only reveal a share of a secret key upon obtaining "proof" of the challenged party's misbehavior.

Looking ahead to our signature collection implementations, the specific form and content of a valid challenge and valid response will vary slightly, but the core challenge-response structure remains the same. To simplify the presentation, we describe a generic challenge subroutine in terms of abstract validation functions (modeled after abstractions of external validity found elsewhere in the consensus literature, e.g. [3, 5]). When we present our signature collection protocols, we will instantiate the generic challenge subroutine with different validation functions to fulfill slightly different purposes. We also define a response selection function that abstracts the logic for selecting a valid response.

To support a variety of use cases, we make only very mild assumptions about the validation functions and selection functions: (1) for any valid challenge, a valid response exists and can be computed given the secret key of the challenged party, and (2) the output of the selection function is a valid response to the given challenge. (Later, we will show these assumptions hold for each concrete instantiation.)

Definition 4.1 (Challenge-Response Family). An *ℓ-bit Challenge-Response Family* $\mathcal{F}$ is specified by the following functions, where challenges and responses are ℓ-bit strings.

- $\mathsf{ChallengeValidation}(c)$: takes a challenge and outputs $b \in \{0,1\}$. c is *valid* if $b = 1$.
- $\mathsf{ResponseValidation}_{pk_i}(c, r)$: takes a challenge/response pair and outputs $b \in \{0,1\}$. If c is valid, r is a *valid response for c from p_i* if $b = 1$.
- $\mathsf{ResponseSelection}_{sk_i}(c, st_i)$: given valid challenge c, secret key sk_i, and additional state information st_i, generates a valid response r for c.

Definition 4.2 (Leader Challenge). Let Π be an n-party protocol with a designated leader p_i, $i \in [n]$, in which p_i is given an input $c \in \{0,1\}^\ell$ and $k \in [n]$, and outputs a value $r \in \{0,1\}^\ell \dot\cup \{\perp\}$. Π is a $\hat{f}$-resilient *Leader Challenge* protocol for ℓ-bit challenge-response family $\mathcal{F}$ if the following properties hold w.r.t. the functions of $\mathcal{F}$.

- **Validity:** If $f \leq \hat{f}$ and c is a valid challenge, then honest p_i outputs a valid response for c.
- **Security:** No secret keys or secret key shares of honest parties are revealed.

In each of our instantiations of the above definitions, we will require responses to be pairs of messages and corresponding signature shares from party p_k. Hence, unforgeability of signatures will imply that only p_k can generate a valid response—unless its secret key is revealed to the leader, which is prohibited if it is honest.

We provide a protocol for leader challenge in Figure 2. Intuitively speaking, communication is kept low by the use of threshold signatures and the mechanism to, if necessary, reveal the key of p_k provided by the model. However, we note that it is crucial to prevent dishonest parties causing many parties p_k to respond to challenges. This is achieved by requiring many honest parties to sign off on k before p_k is forced to respond. The high threshold of $\lceil 3n/4 \rceil$ easily limits this to a single party, at the expense of reliably extracting a response from p_k only if $f \leq \lfloor n/4 \rfloor$.

Proving the protocol correct is straightforward.

Lemma 4.3. *$\Pi_{LC}^{L,\mathcal{F}}$ satisfies the security property of Definition 4.2.*

Proof. No keys are ever revealed by honest parties. The only step where an honest party might reveal a secret share for a key is Step 6, and only if it did not receive a valid response from the party p_k it forward a valid challenge from p_i to. But if p_k is honest, it responded with a valid response to all parties in the previous step. Thus, no honest party ever reveals a share of a secret key belonging to an honest party. □

Lemma 4.4. *$\Pi_{LC}^{L,\mathcal{F}}$ satisfies the validity property of Definition 4.2 for $\hat{f} = \lfloor n/4 \rfloor$.*

$\Pi_{\mathbf{LC}}^{L,\mathcal{F}}$: Leader Challenge Protocol for Challenge-Response Family $\mathcal{F}$

Parameters: leader p_i.
Input: p_i inputs (k, c), where p_k is the challenged party and c is the challenge.
Main: Parties execute the following steps, proceeding in synchronous rounds:

(1) p_i sends k to all parties.
(2) Each party p_j, $j \in [n]$, responds with $\langle k \rangle_j^{\lceil 3n/4 \rceil}$ to p_i.
(3) If p_i received enough shares to do so, it computes $\langle k \rangle^{\lceil 3n/4 \rceil}$ and sends $(\langle k \rangle^{\lceil 3n/4 \rceil}, c)$ to all parties.
(4) If party p_j, $j \in [n]$, receives $(\langle k \rangle^{\lceil 3n/4 \rceil}, c)$ with valid c, it forwards it to party p_k.
(5) If party p_k, $k \in [n]$, receives $(\langle k \rangle^{\lceil 3n/4 \rceil}, c)$ with valid c from party p_j, $j \in [n]$, it responds with $\mathsf{ResponseSelection}_{sk_k}(c, st_k)$.
(6) If p_j, $j \in [n]$, sent $(\langle k \rangle^{\lceil 3n/4 \rceil}, c)$ to p_k in Step 4, it checks whether it received a valid response r from p_k. If yes, it forwards r to p_i. If no, it sends its share of the secret key of p_k to p_i.
(7) If p_i receives any r valid for c, it outputs one of them. Otherwise, if it received enough shares of the secret key of p_k, it reconstructs it, uses it to create any r valid for c (using an arbitrary value for st_k), and outputs that. Otherwise, it outputs $\perp$.

Fig. 2. A generic leader challenge protocol.

Proof. If p_i is honest and there are no more than $f \leq \lfloor n/4 \rfloor$ dishonest parties, it receives enough signature shares to compute $\langle k \rangle^{\lceil 3n/4 \rceil}$ in Step 3 and send $(\langle k \rangle^{\lceil 3n/4 \rceil}, c)$ to all parties. If c is valid, all honest parties forward this message to p_k. If p_k responds to any honest party with a valid response r for c, they forward it to p_i, which outputs it. Otherwise, all of them send their share of the secrete key of p_k to p_i. In this case, p_i reconstructs the key of p_k, generates a valid response r for c, and outputs it. □

Theorem 4.5. *$\Pi_{LC}^{L,\mathcal{F}}$ solves $\lceil 3n/4 \rceil$-resilient leader challenge (Definition 4.2) within 6 rounds using $O(n(\ell + \kappa))$ bits.*

Proof. It can be readily verified from the description of the protocol that 6 communication rounds are required and that all messages contain at most one challenge or response, at most one additional signature, and one index k (of $O(\log n) \subseteq O(\kappa)$ bits). Apart from the initial signature share sent to p_i, honest parties other than p_i communicate only when receiving a valid $\lceil 3n/4 \rceil$-threshold signature on an index k. However, they create only a single share and there are $t < n/2$ dishonest parties, so that at most one such threshold signature is created. Thus, honest parties other than p_i and some unique p_k send messages to p_i and p_k only. The bit complexity bound follows.

The validity and security properties follow from Lemmas 4.3 and 4.4. □

4.2 Signature Collection for Binary Inputs

Signature collection for inputs is the only subroutine that is more straightforward for the binary case, so we present it for this special case. Recall that our goal is to obtain a signature $\langle b \rangle^{\lceil 3n/4 \rceil}$ for some $b \in \{0, 1\}$ such that (i) b is unique and (ii) some honest party had input $b_i = b$. Formally:

Definition 3.1 (Signature Collection for Inputs). Given an input $\mathrm{op}_i \in \{0,1\}^\ell$ at each party p_i, $i \in [n]$, *$\hat{f}$-resilient Signature Collection for Inputs with Leader Set $L \subseteq [n]$* has each p_i compute an output s_i that is either a valid threshold-$\lceil 3n/4 \rceil$ signature on some $v \in \{0,1\}^\ell$ or $\perp$, where the following properties hold.

- **No equivocation:** There is at most one $v \in \{0,1\}^\ell$ such that signature $\langle v \rangle^{\lceil 3n/4 \rceil}$ is created.
- **Validity:** If $\langle v \rangle^{\lceil 3n/4 \rceil}$ is created for some $v \in \{0,1\}^\ell$, then $\text{op}_i = v$ for some honest party p_i.
- **Liveness:** If $f \leq \hat{f}$, then all parties output $\langle v \rangle^{\lceil 3n/4 \rceil}$ for some $v \in \{0,1\}^\ell$.

Π^L_{SCI}: Signature Collection for Inputs

Parameters: estimated corruption threshold $\hat{f}$, designated leader set L with $|L| = 2\hat{f} + 1$, and challenge-response family $\mathcal{F}_{\text{SCI}}$ (as defined below)

Input: Each party p_i holds an input $\text{op}_i \in \{0,1\}$.

Main: Parties execute the following steps, proceeding in synchronous rounds:

(1) Each party p_j computes $r_j = \text{SCI.ResponseSelection}_{sk_j}(0^L, \text{op}_j) = \langle \text{op}_j \rangle_j^{t+1}$ and sends it to all parties in L.

(2) Sequentially for each $i \in L$:

  (a) Each party p_j, $j \in L$ does: if there is a party p_k for which p_j has not received $\langle b \rangle_k^{t+1}$ for some $b \in \{0,1\}$ yet, p_j selects the lowest such index $k \in [n]$ and sends k to the iteration leader p_i. If p_i holds such a signature share, it sends it to p_j.

  (b) If there is a party p_k for which the iteration leader p_i has no $\langle b \rangle_k^{t+1}$ for some $b \in \{0,1\}$, p_i inputs the lowest such index $k \in [n]$ with challenge 0^L to the leader challenge subroutine for challenge-response family $\mathcal{F}_{\text{SCI}}$. (All parties participate with leader p_i.)

  (c) If p_i outputs $\langle b \rangle_k^{t+1}$ for some $b \in \{0,1\}$, it sends it to each p_j, $j \in L$.

(3) Each p_i, $i \in L$ does: if p_i holds $t+1$ shares $\langle b \rangle_k^{t+1}$ for some $b \in \{0,1\}$, it computes $\langle b \rangle^{t+1}$ and sends it to each party in L.

(4) The parties in L run ($|L|$-party binary) BA, where $p_i \in L$ uses input 0 iff it received $\langle 0 \rangle^{t+1}$ in the previous step and input 1 otherwise.

(5) If p_i, $i \in L$, outputs o_i from BA and previously received $\langle o_i \rangle^{t+1}$, it sends it to all parties.

(6) Each party p_j, $j \in [n]$ does: if there is a $b \in \{0,1\}$ such that it received $\langle b \rangle^{t+1}$ from a majority of parties in L, it sends $\langle b \rangle_j^{\lceil 3n/4 \rceil}$ to all parties in L.

(7) If p_i, $i \in L$, received message $\langle b \rangle_j^{\lceil 3n/4 \rceil}$ from $\lceil 3n/4 \rceil$ parties p_j for some $b \in \{0,1\}$, it computes $\langle b \rangle^{\lceil 3n/4 \rceil}$ and sends it to all parties.

(8) If p_j, $j \in [n]$, received $\langle b \rangle^{\lceil 3n/4 \rceil}$ for some $b \in \{0,1\}$ from a majority of parties in L, it outputs it. Otherwise, it outputs $\perp$.

$\text{SCI.ChallengeValidation}(c)$ is constant 1, i.e., any challenge is valid.

$\text{SCI.ResponseValidation}_{pk_k}(c, r)$: r is valid if and only if $r = \langle b \rangle_k^{t+1}$ for some $b \in \{0,1\}$.

$\text{SCI.ResponseSelection}_{sk_k}(c, \text{op}_k)$: returns $\langle \text{op}_k \rangle_k^{t+1}$.

Fig. 3. An algorithm for Signature Collection for Inputs with designated leader set L.

Observe that in the binary case, forcing all parties to provide a signature share for their input bit means that the majority value b has multiplicity at least $\lceil n/2 \rceil > t$. Thus, doing so enables us to generate a proof that adopting b is safe. Afterwards, this proof can be used to ask parties for a signature share for $\langle b \rangle^{\lceil 3n/4 \rceil}$. Honest parties will oblige, but only for a single value. This implies that only the value b for which the majority of honest parties sign can see the threshold signature $\langle b \rangle^{\lceil 3n/4 \rceil}$ being created. On the other hand, if the leader set L has an honest majority, this majority will agree on the value b it asks for, and if f is sufficiently small, enough shares will be obtained to create the threshold signature. This approach is implemented in Figure 3.

Before turning to proving the protocol correct, let us discuss a suitable choice of challenge-response family. In this case, the structure is quite simple: all that is required is for each party to

provide a signature share for *some* value, where honest parties stick to their actual inputs. This means that we do not actually need to make use of the challenge part of the functionality.

Concretely, we declare any challenge valid and specify a valid response as a threshold-$(t+1)$ signature share on some value $b \in \{0,1\}$. The response selection function then has party p_k generate such a signature share with $b = \mathrm{op}_k$.

Lemma 4.6. *$\mathcal{F}_{SCI}$ is an $O(\kappa)$-bit challenge-response family.*

Proof. Recall that it is required that (1) for any valid challenge, a valid response exists, and (2) given inputs of the expected form, the response selection function outputs a valid response. First, note that a valid response does exist, because a valid threshold $(t+1)$ signature share on a bit b under the challenged party's secret key is a valid response. Furthermore, any party who holds the secret key of p_k can compute a signature share on behalf of p_k on arbitrary values, and an honest party only calls the response selection function in situations where they hold the secret key, so the response selection function does produce a valid response.

Since the responses contain a signature share on a (short) message, we can fix $\ell = O(\kappa)$. □

We now show that the protocol satisfies the required properties.

Lemma 4.7. *Π^L_{SCI} satisfies the no equivocation property of signature collection for inputs.*

Proof. Observe that each honest party p_j will generate a message of form $\langle b \rangle_j^{\lceil 3n/4 \rceil}$ only for a single value b. Furthermore, security of leader challenge guarantees that no information on honest party's keys is revealed. Thus, security of the secret sharing scheme for the keys and unforgeability of signatures implies that the adversary cannot forge signature shares obtain signature shares of honest parties by different means. Thus, there is one $b \in \{0,1\}$ for which at most half of the honest parties provide a share, i.e., at most $t + (n-t)/2 = (n+t)/2 < 3n/4$ shares can be generated. Therefore, by the security of the threshold signature scheme, $\langle b \rangle^{\lceil 3n/4 \rceil}$ cannot be created for this b. □

Lemma 4.8. *Π^L_{SCI} satisfies the validity property of signature collection for inputs.*

Proof. Suppose that there is a bit b such that no honest party p_j has input $\mathrm{op}_j = b$. Then no honest party p_j generates a signature share $\langle b \rangle_j^{t+1}$ (including during the leader challenge protocol). Furthermore, security of leader challenge guarantees that no information on honest party's keys is revealed. Thus, security of the secret sharing scheme for the keys and unforgeability of signatures implies that the adversary cannot forge signature shares obtain signature shares of honest parties by different means. Therefore, by the security of the threshold signature scheme $\langle b \rangle^{\lceil 3n/4 \rceil}$ cannot be created. □

Lemma 4.9. *Provided that $f \leq \hat{f} \leq \lfloor n/4 \rfloor$ and the leader challenge subroutine has resilience $\hat{f}$, Π^L_{SCI} satisfies the liveness property for resilience $\hat{f}$.*

Proof. After the first step, each honest party in L has received $\langle \mathrm{op}_j \rangle_j^{(t+1)}$ from each honest party p_j. For an iteration i of the loop, let k be minimal with the property that some honest leader $p_j \in L$ has not received $\langle b \rangle_k^{(t+1)}$ for some $b \in \{0,1\}$ from some (dishonest) party p_k. Suppose that the iteration leader p_i is honest. We distinguish two cases:

- If p_i has already received $\langle b \rangle_k^{(t+1)}$ for some $b \in \{0,1\}$ from p_k, each $p_j \in L$ that does not will query p_i for index k and receive $\langle b \rangle_k^{(t+1)}$.

- If p_i has not yet received $\langle b \rangle_k^{(t+1)}$ for some $b \in \{0,1\}$ from p_k, it calls the challenge subroutine for index k. If $f \leq \hat{f}$, the subroutine outputs $\langle b \rangle_k^{(t+1)}$ at p_i for some $b \in \{0,1\}$, which p_i sends to all parties in L.

Either way, if $f \leq \hat{f}$, in each iteration with an honest leader p_i, the minimal index k such that some honest leader $p_j \in L$ has not received $\langle b \rangle_k^{(t+1)}$ for some $b \in \{0,1\}$ yet must increase. If $f \leq \hat{f}$, there are at most f such indices k and at least $2\hat{f} + 1 - f > f$ honest leaders. We conclude that honest leaders obtain signature shares from all parties.

Next, if $f \leq \hat{f} < |L|/2$, the BA instance in Step 4 succeeds, and Steps 4–5 ensure that each honest party observes a majority of leaders send the same value of b in Step 6. Hence, each honest party p_j generates a share $\langle b \rangle_j^{\lceil 3n/4 \rceil}$ on the same value b and sends it to all leaders. Since there are $n - f \geq n - \hat{f} \geq n - \lfloor n/4 \rfloor = \lceil 3n/4 \rceil$ honest parties, honest leaders compute and transmit $\langle b \rangle^{\lceil 3n/4 \rceil}$ to all parties. Thus, each honest party outputs $\langle b \rangle^{\lceil 3n/4 \rceil}$ (by Lemma 4.7 for unique b). □

Theorem 4.10. *Provided the challenge-response subroutine has resilience $\hat{f} \leq \lfloor n/4 \rfloor$, Π_{SCI}^L solves signature collection for inputs with resilience $\hat{f}$. If R_{CR} and B_{CR} are the round and bit complexities of the challenge-response subroutine (with $\ell = O(\kappa)$) and R_{BA} and B_{BA} the round and bit complexities of the BA subroutine, respectively, this requires $O((\hat{f}+1)(R_{CR}+1) + R_{BA})$ rounds and $O((\hat{f}+1)(B_{CR} + n\kappa) + B_{BA})$ bits.*

Proof. Correctness follows from Lemmas 4.6 to 4.9.

Regarding complexities, observe that there are $|L| = O(\hat{f} + 1)$ iterations of the loop. Each comprises one invocation of the leader challenge subroutine with $\ell = O(\kappa)$ and $O(1)$ rounds of message exchange between p_i and other parties, where each message comprises $O(\kappa)$ bits. The remainder of the protocol calls BA on node set L once and otherwise performs $O(1)$ rounds of message exchange between parties in L and other parties, where each message comprises $O(\kappa)$ bits. □

Corollary 4.11. *For $\hat{f} \leq \lfloor n/4 \rfloor$, signature collection for inputs can be solved in $O(\hat{f} + 1)$ rounds with $O(n(\hat{f} + 1)\kappa)$ bits.*

Proof. We apply Theorem 4.10, where the leader challenge subroutine is given by Theorem 4.5 and the BA subroutine is given by the threshold-signature based BA protocol of Momose and Ren [23], which requires $O(|L|) = O(\hat{f})$ rounds and has bit complexity $O(|L|^2\kappa) = O(n(\hat{f}+1)\kappa)$. □

4.3 Signature Collection for Decision

We present our second signature collection implementation, signature collection for decision (SCD). Like the SCI subroutine, the SCD subroutine (shown in Figure 4) is run among n parties, with a designated leader set $L \subset [n]$. Each party $p_i \in [n]$ has input that is either $\perp$ or a threshold signature $\langle v \rangle^{\lceil 3n/4 \rceil}$ for a value $v \in \{0,1\}^\ell$, and outputs either $\perp$ or $\langle v \rangle^{\lceil n \rceil}$ for some $v \in \{0,1\}^\ell$. Formally:

Definition 3.2 (Signature Collection for Decision). $\hat{f}$-resilient Signature Collection for Decision with Leader Set $L \subseteq [n]$ is defined as follows. Each party p_i, $i \in [n]$, has input $\langle v \rangle^{\lceil 3n/4 \rceil}$ for some $v \in \{0,1\}^\ell$ or $\perp$ and computes an output s_i that is either $\langle v \rangle^n$ for some $v \in \{0,1\}^\ell$ or $\perp$, where the following properties hold.

- **Validity:** If $\langle v \rangle^n$ is created for some $v \in \{0,1\}^\ell$, then each honest party received a threshold-$\lceil 3n/4 \rceil$ signature for (b, commit).
- **Liveness:** If $f \leq \hat{f}$ parties are corrupted, all honest parties receive input $\langle v \rangle^{\lceil 3n/4 \rceil}$ for some $v \in \{0,1\}^\ell$, and $\langle v' \rangle^{\lceil 3n/4 \rceil}$ cannot be generated for $v' \neq v$, then honest parties output $\langle v \rangle^n$.

In the full early-stopping BA protocol, the inputs to this subroutine are generated using the signature collection for inputs subroutine, so we will be certain that v is unique. The decision routine makes *all* parties partake in the signing process, so that if a threshold signature is generated, we can be sure that all parties received $\langle v \rangle^{\lceil 3n/4 \rceil}$—even if the leader set is completely corrupted. We then have each signing party adopt the value it signed as their current preference, so that creation of a decision signature renders the system univalent, i.e., *decides* on v.

Our SCD implementation follows a very similar pattern to the SCI implementation. It also makes use of an instantiation of the leader challenge protocol to coerce a signature from (or on behalf of) uncooperative parties. We prevent equivocation by leveraging that the signature collection for inputs subroutine guarantees that only for a single value v, a signature $\langle v \rangle^{\lceil 3n/4 \rceil}$ can be created. We use this signature as prompt for p_i to provide a signature share $\langle v \rangle_i^n$ and adopt value v. We can safely use it as challenge for the leader challenge subroutine, preventing $f \leq \hat{f}$ corrupted parties from blocking progress. We remark that parties do not generate on own state information when generating their responses, which we represent by using the empty word ε as second argument of SCD.ResponseSelection.

Before continuing, we prove that the challenge-response family used in this subroutine (formally defined in Figure 4) meets the required criteria.

Lemma 4.12. *$\mathcal{F}_{SCD}$ is an $O(\kappa + \ell)$-bit challenge-response family.*

Proof. Recall that is is required that (1) for any valid challenge, a valid response exists, and (2) given inputs of the expected form, the response selection function outputs a valid response. First, note that a valid response does exist, because a valid threshold-n signature share on v under the challenged party's secret key is a valid response to challenge $\langle v \rangle^{\lceil 3n/4 \rceil}$. Furthermore, any party who holds the secret key of p_k can compute a signature share on behalf of p_k on arbitrary values, and an honest party only calls the response selection function in situations where they hold the secret key, so the response selection function does produce a valid response.

Since challenges and responses consist of signatures or signature shares on ℓ-bit messages (where ℓ is from Definition 2.2, we have that $\mathcal{F}_{\text{SCD}}$ is an $O(\kappa + \ell)$-bit challenge-response family. □

Lemma 4.13. *Π_{SCD}^L satisfies the validity condition of signature collection for decision.*

Proof. For validity, observe that an honest party p_j will only create and send $\langle v \rangle_j^n$ if it received $\langle v \rangle^{\lceil 3n/4 \rceil}$, either as input or as challenge. Furthermore, security of the leader challenge subroutine ensures that no honest party ever reveals a secret share of an honest party's secret key (or its key). Thus, if $\langle v \rangle^n$ is created, each honest party must have received $\langle v \rangle^{\lceil 3n/4 \rceil}$. □

Lemma 4.14. *Π_{SCD}^L satisfies the liveness condition of signature collection for decision.*

Proof. Suppose that all honest parties receive input $\langle v \rangle^{\lceil 3n/4 \rceil}$. Then each party in L receives $\langle v \rangle_j^n$ from each honest party p_j. Analogously to the proof of Lemma 4.9, if $f \leq \hat{f}$, honest leaders obtain signature shares from all parties during the loop.

Hence, honest parties in L will compute $\langle v \rangle^n$ and send it to all parties. Note that if $\langle v' \rangle^{\lceil 3n/4 \rceil}$ is not created for $v' \neq v$, the same applies to $\langle v' \rangle^n$, as honest party p_j then cannot receive a valid challenge prompting it to provide a share $\langle v' \rangle_j^n$. The claim of the lemma follows. □

Theorem 4.15. *Provided the challenge-response subroutine has resilience $\hat{f} \leq \lfloor n/4 \rfloor$, Π_{SCD}^L solves signature collection for decision with resilience $\hat{f}$. If R_{CR} and B_{CR} are the round and bit complexities of the challenge-response subroutine, respectively, this requires $O((\hat{f}+1)(R_{CR}+1))$ rounds and $O((\hat{f}+1)(B_{CR}+n(\kappa+\ell)))$ bits.*

$\Pi^{L}_{\mathbf{SCD}}$: Signature Collection for Decision

Parameters: estimated corruption threshold $\hat{f}$, designated leader set L with $|L| = 2\hat{f} + 1$, and challenge-response family $\mathcal{F}_{\mathrm{SCD}}$ (as defined below).

Input: Party p_i's receives input $\sigma_i = \langle v \rangle^{\lceil 3n/4 \rceil}$ for some $v \in \{0,1\}^\ell$ or $\sigma_i = \perp$.

Main: Parties execute the following steps, proceeding in synchronous rounds:

(1) If party p_i received input $\langle v \rangle^{\lceil 3n/4 \rceil}$, it computes $\langle v \rangle^n_i$ and sends it to all parties in L.

(2) Sequentially for $i \in L$:

(a) Each party p_j, $j \in L$ does: if there is a party p_k for which p_j has not received $\langle v \rangle^n_i$ for some $v \in \{0,1\}^\ell$ yet, p_j selects the lowest such index $k \in [n]$ (if any) and sends k to the iteration leader p_i. If p_i holds such a share, it sends it to p_j.

(b) If there is a party p_k for which the iteration leader p_i has no $\langle v \rangle^n_i$ for some $v \in \{0,1\}^\ell$ yet and $\sigma_i \neq \perp$, it inputs the lowest such index $k \in [n]$ with challenge σ_i to the leader challenge subroutine for challenge-response family $\mathcal{F}_{\mathrm{SCD}}$. (All parties participate with leader p_i.)

(c) If p_i outputs $\langle v \rangle^n_i$ for some $v \in \{0,1\}^\ell$, it sends it to each p_j, $j \in L$.

(3) For each $i \in L$, if there is a $v \in \{0,1\}^\ell$ such that p_i holds sufficiently many shares $\langle v \rangle^n_i$ to compute $\langle v \rangle^n$, it does so and sends it to all parties.

(4) If p_j, $j \in [n]$, received $\langle v \rangle^n$ for some $v \in \{0,1\}^\ell$, it outputs one; otherwise it outputs $\perp$.

$\mathsf{SCD.ChallengeValidation}(c)$**:** c is valid if and only if it is valid $\langle v \rangle^{\lceil 3n/4 \rceil}$ for some $v \in \{0,1\}^\ell$.

$\mathsf{SCD.ResponseValidation}_{pk_k}(c, r)$**:** returns 1 (i.e., valid), if and only if $c = \langle v \rangle^{\lceil 3n/4 \rceil}$ and $r = \langle v \rangle^n_k$ for some $v \in \{0,1\}^\ell$; otherwise, return 0.

$\mathsf{SCD.ResponseSelection}_{sk_k}(c, \varepsilon)$**:** if $c = \langle v \rangle^{\lceil 3n/4 \rceil}$, output $\langle v \rangle^n_k$ and otherwise 0^ℓ.

Fig. 4. An algorithm for signature collection for decision with designated leader set L.

Proof. Correctness follows from Lemmas 4.12 to 4.14.

Regarding complexities, observe that there are $|L| = O(\hat{f} + 1)$ iterations of the loop. Each comprises one invocation of the leader challenge subroutine with an $O(\kappa + \ell)$-bit family and $O(1)$ rounds of message exchange between p_i and other parties, where each message comprises $O(\kappa + \ell)$ bits. The remainder of the protocol has parties in L send one $O(\kappa + \ell)$-bit message to each party. □

Corollary 4.16. *For $\hat{f} \leq \lfloor n/4 \rfloor$ and $\ell \in O(\kappa)$, signature collection for decision can be solved in $O(\hat{f} + 1)$ rounds with $O(n(\hat{f} + 1)\kappa)$ bits.*

REFERENCES

[1] Ittai Abraham, T.-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. 2019. Communication Complexity of Byzantine Agreement, Revisited. In *Symposium on Principles of Distributed Computing (PODC)*, Peter Robinson and Faith Ellen (Eds.). ACM, 317–326.

[2] Ittai Abraham and Danny Dolev. 2015. Byzantine Agreement with Optimal Early Stopping, Optimal Resilience and Polynomial Complexity. In *47th ACM STOC*, Rocco A. Servedio and Ronitt Rubinfeld (Eds.). ACM Press, Portland, OR, USA, 605–614. https://doi.org/10.1145/2746539.2746581

[3] Ittai Abraham, Dahlia Malkhi, and Alexander Spiegelman. 2019. Asymptotically Optimal Validated Asynchronous Byzantine Agreement. In *38th ACM PODC*, Peter Robinson and Faith Ellen (Eds.). ACM, Toronto, ON, Canada, 337–346. https://doi.org/10.1145/3293611.3331612

[4] Erica Blum, Jonathan Katz, Chen-Da Liu-Zhang, and Julian Loss. 2020. Asynchronous Byzantine Agreement with Subquadratic Communication. In *TCC 2020, Part I (LNCS, Vol. 12550)*, Rafael Pass and Krzysztof Pietrzak (Eds.). Springer, Cham, Switzerland, Durham, NC, USA, 353–380. https://doi.org/10.1007/978-3-030-64375-1_13

[5] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. 2001. Secure and Efficient Asynchronous Broadcast Protocols. In *CRYPTO 2001 (LNCS, Vol. 2139)*, Joe Kilian (Ed.). Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA, 524–541. https://doi.org/10.1007/3-540-44647-8_31

[6] Michael Capalbo, Omer Reingold, Salil Vadhan, and Avi Wigderson. 2002. Randomness conductors and constant-degree lossless expanders. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing* (Montreal, Quebec, Canada) *(STOC '02)*. Association for Computing Machinery, New York, NY, USA, 659–668. https://doi.org/10.1145/509907.510003

[7] Benny Chor, Michael Merritt, and David B. Shmoys. 1989. Simple constant-time consensus protocols in realistic failure models. *J. ACM* 36, 3 (July 1989), 591–614.

[8] Pierre Civit, Muhammad Ayaz Dzulfikar, Seth Gilbert, Rachid Guerraoui, Jovan Komatovic, and Manuel Vidigueira. 2024. DARE to Agree: Byzantine Agreement With Optimal Resilience and Adaptive Communication. In *43rd ACM PODC*, Ran Gelles, Dennis Olivetti, and Petr Kuznetsov (Eds.). ACM, Nantes, France, 145–156. https://doi.org/10.1145/3662158.3662792

[9] Shir Cohen, Idit Keidar, and Alexander Spiegelman. 2020. Brief Announcement: Not a COINcidence: Sub-Quadratic Asynchronous Byzantine Agreement WHP. In *39th ACM PODC*, Yuval Emek and Christian Cachin (Eds.). ACM, Virtual Event, Italy, 175–177. https://doi.org/10.1145/3382734.3405708

[10] Shir Cohen, Idit Keidar, and Alexander Spiegelman. 2022. Brief Announcement: Make Every Word Count: Adaptive Byzantine Agreement with Fewer Words. In *41st ACM PODC*, Alessia Milani and Philipp Woelfel (Eds.). ACM, Salerno, Italy, 421–423. https://doi.org/10.1145/3519270.3538458

[11] Andrei Constantinescu, Marc Dufay, Anton Paramonov, and Roger Wattenhofer. 2025. From Few to Many Faults: Adaptive Byzantine Agreement with Optimal Communication. arXiv:2505.19989 [cs.DC] https://arxiv.org/abs/2505.19989

[12] Danny Dolev and Rüdiger Reischuk. 1982. Bounds on Information Exchange for Byzantine Agreement. In *1st ACM PODC*, Robert L. Probert, Michael J. Fischer, and Nicola Santoro (Eds.). ACM, Ottawa, Canada, 132–140. https://doi.org/10.1145/800220.806690

[13] Danny Dolev, Ruediger Reischuk, and H. Raymond Strong. 1982. 'Eventual' is Earlier than 'Immediate'. In *Symposium on Foundations of Computer Science (SFCS)*. 196–203.

[14] D. Dolev and H. R. Strong. 1983. Authenticated Algorithms for Byzantine Agreement. *SIAM J. Comput.* 12, 4 (1983), 656–666. https://doi.org/10.1137/0212045 arXiv:https://doi.org/10.1137/0212045

[15] Fatima Elsheimy, Simon Holmgaard Kamp, and Julian Loss. 2025. Early Stopping is Cheap. Cryptology ePrint Archive, Paper 2025/1107. https://eprint.iacr.org/2025/1107

[16] Fatima Elsheimy, Julian Loss, and Charalampos Papamanthou. 2024. Early Stopping Byzantine Agreement in $(1+\epsilon)\cdot f$ Rounds. In *ASIACRYPT 2024, Part VI (LNCS, Vol. 15489)*, Kai-Min Chung and Yu Sasaki (Eds.). Springer, Singapore, Singapore, Kolkata, India, 398–424. https://doi.org/10.1007/978-981-96-0938-3_13

[17] Fatima Elsheimy, Julian Loss, and Charalampos Papamanthou. 2025. Towards Optimal Early Stopping Agreement Protocols. Cryptology ePrint Archive, Paper 2025/246. https://eprint.iacr.org/2025/246

[18] Fatima Elsheimy, Giorgos Tsimos, and Charalampos Papamanthou. 2024. Deterministic Byzantine Agreement with Adaptive $O(n \cdot f)$ Communication. In *35th SODA*, David P. Woodruff (Ed.). ACM-SIAM, Alexandria, VA, USA, 1120–1146. https://doi.org/10.1137/1.9781611977912.43

[19] Matthias Fitzi, Chen-Da Liu-Zhang, and Julian Loss. 2021. A New Way to Achieve Round-Efficient Byzantine Agreement. In *40th ACM PODC*, Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen (Eds.). ACM, Virtual Event, Italy, 355–362. https://doi.org/10.1145/3465084.3467907

[20] Diana Ghinea, Vipul Goyal, and Chen-Da Liu-Zhang. 2022. Round-Optimal Byzantine Agreement. In *Advances in Cryptology – EUROCRYPT 2022*, Orr Dunkelman and Stefan Dziembowski (Eds.). Springer International Publishing, 96–119.

[21] Anna Rochelle Karlin and Andrew Chi-Chih Yao. 1984. Probabilistic lower bounds for byzantine agreement and clock synchronization. Unpublished manuscript.

[22] Christoph Lenzen and Joel Rybicki. 2019. Near-optimal self-stabilising counting and firing squads. *Distrib. Comput.* 32, 4 (Aug. 2019), 339–360. https://doi.org/10.1007/s00446-018-0342-6

[23] Atsuki Momose and Ling Ren. 2021. Optimal Communication Complexity of Authenticated Byzantine Agreement. In *35th International Symposium on Distributed Computing (DISC 2021) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 209)*, Seth Gilbert (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 32:1–32:16. https://doi.org/10.4230/LIPIcs.DISC.2021.32

[24] M. Pease, R. Shostak, and L. Lamport. 1980. Reaching Agreement in the Presence of Faults. *J. ACM* 27, 2 (April 1980), 228–234.

[25] Kenneth J Perry and Sam Toueg. 1984. *An authenticated byzantine generals algorithm with early stopping.* Technical Report. Cornell University.

[26] Alexander Spiegelman. 2021. In Search for an Optimal Authenticated Byzantine Agreement. In *35th International Symposium on Distributed Computing (DISC 2021) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 209)*, Seth Gilbert (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 38:1–38:19. https://doi.org/10.4230/LIPIcs.DISC.2021.38

A SILENT BYZANTINE AGREEMENT

We modularize the construction from [22] via an intermediate routine, *Silent k-Graded Consensus*. It is a close relative of graded consensus (readers familiar with Proxcensus [19] will find it quite similar, with slightly different syntax) with an additional silence property.

Definition A.1 (Silent k-Graded Consensus). Let Π be an n-party protocol in which each party p_i holds an input $b_i \in \{0,1\}$ and outputs a value $g_i \in \{0,1,\ldots,k\}$. Π is a *t-resilient silent k-graded consensus protocol* if it guarantees the following properties in the presence of up to t corruptions:

- **Validity:** If $b_i = b$ for each honest party p_i, then each honest party p_i outputs grade kb.
- **Graded Agreement:** For any two honest parties p_i, p_j, $|o_i - o_j| \leq 1$.
- **Silence:** If $b_i = 0$ for each honest party p_i, no honest party p_i sends any message.

Silent k-Graded Consensus

Input: Each party p_i holds input $b_i \in \{0,1\}$.
Main: Each party p_i executes the following steps:
(1) Initialize $g_i = b_i$.
(2) For $g = 1,\ldots,k$, sequentially, do the following:
  (a) If $g_i = g$, send $\langle g \rangle_i^{t+1}$ to all parties.
  (b) If there is a subset $S \in [n]$ of size at least $t+1$ such that for each $j \in S$, $\langle g \rangle_j^{t+1}$ was received from p_j, set $g_i := g+1$, compute $\langle g \rangle^{t+1}$, and send it to all parties.
  (c) If $\langle g \rangle^{t+1}$ is received from any party, set $g_i := \max\{g, g_i\}$.
(3) Output g_i.

Fig. 5. An algorithm for silent k-graded consensus.

Theorem A.2. *The algorithm in Definition A.1 solves silent k-graded consensus in $2k$ rounds with bit complexity $O(kn^2\kappa)$.*

Proof. The round and bit complexity bounds can be readily verified from the description of the algorithm.

If each honest party has input $b_i = 0$, none of them create any signature shares in the first round. Thus, none of them receives a valid threshold signature, and this property is maintained inductively for all iterations. Thus, no honest party sends any message, proving silence, and all honest parties output grade 0.

To prove validity, it remains to consider the case in which all honest parties have input 1. Thus, all $n - t > t$ honest parties create and transmit a signature share in the first round and create the corresponding threshold signature. By induction, these parties keep creating shares and signatures in all iterations, resulting in an output grade of k at all honest parties, proving validity.

It remains to show graded agreement. Let g be the maximum with the property that an honest party p_i executed Step 2a (if no such g exists, all honest parties have input 0 and the graded agreement follows from validity). By unforgeability of signatures, no message $\langle g' \rangle^{t+1}$ can be created, implying that no honest party can execute Step 2b for $g' \geq g$ and no honest party can execute Step 2c for

$g' > g$. Thus, no honest party outputs a grade larger than g. If $g = 1$, graded agreement thus holds. If $g > 1$, g_i must have executed Step 2b in loop iteration $g - 1$. Hence, it computed $\langle g-1 \rangle^{t+1}$ and sent it to all parties, prompting honest party p_j to set g_j to a value of at least $g - 1$. As later loop iterations cannot reduce g_j, graded agreement follows. □

$\Pi^s_{\mathbf{shh}}$: Silent Consensus

Parameters: A silent input s, a BA subroutine of round complexity $R(\Pi_{\mathrm{BA}})$ with communication bound $C(\Pi_{\mathrm{BA}})$ for each party (such that n times $C(\Pi_{\mathrm{BA}})$ is its bit complexity), and a silent 4-graded consensus subroutine.

Input: Each party holds input $v_i \in \{0,1\}^\ell$.

Main: Each party p_i executes the following steps:

(1) Run silent 4-graded consensus with input $b_i = 0$ if $v_i = s$ and $b_i = 1$ else.

(2) Upon receiving output $g_i \in \{0,1,2,3\}$ from the silent 4-graded consensus subroutine:
- If $g_i = 0$: output s and terminate.
- If $g_i = 1$ or $g_i = 2$: run BA with input s.
- If $g_i = 3$: run BA using input v_i.

At any point while running BA, if p_i's communication or running time for the subroutine exceeds $C(\Pi_{\mathrm{BA}})$ bits or $R(\Pi_{\mathrm{BA}})$ rounds, respectively, or if p_i's behavior in Π_{BA} is not well-defined (e.g., because insufficient messages were received), it locally terminates the BA instance.

(3) Upon terminating the BA instance, determine output as follows and then terminate:
- If $g_i = 1$: output s.
- If $g_i \geq 2$: output the local output of the BA instance.

Fig. 6. An algorithm for silent consensus that is silent on input s, given subroutines for silent 4-graded consensus and standard (non-silent) BA.

Theorem A.3. *The algorithm in Figure 6 solves silent consensus. Its round and bit complexities are given by the sum of the complexities of the BA and silent 4-graded consensus routines.*

Proof. The bounds on round and bit complexity follow immediately from the protocol description, nothing that any "problematic" behavior of the BA subroutine arising from not all honest parties participating is captured by terminating if there is unspecified behavior or complexity bounds would be violated.

For the silence property, suppose $v_i = s$ for each honest party p_i. Then by silence of silent k-graded consensus, no honest party sends any messages during the execution of the silent graded consensus. Furthermore, by validity of silent graded consensus, each honest party p_i receives output 0 from the graded consensus and therefore outputs s without executing the BA subroutine.

Now consider the validity property. Suppose each honest party inputs the same input $v_i = v \in \{0,1\}^\ell$. The case that $v = s$ has already been covered, so suppose that $v \neq s$. Thus, all honest parties use input 1 to the silent k-graded consensus subroutine. By validity of this subroutine, all honest parties receive output 3 from silent graded consensus. Thus, each honest party runs BA with input $v_i = v$, which by validity of BA results in output v at all honest parties. Due to having grade 3, this means they also output v from the silent consensus protocol.

Finally, for the consistency property we distinguish three cases, based on the maximum grade $g \in \{0,1,2,3\}$ among honest parties' outputs from the silent 4-graded consensus subroutine:

- $g \leq 1$: all parties output s, regardless of the behavior of the BA subroutine.

- $g \geq 2$: by graded agreement, each honest party p_i has $g_i \geq 1$ and participates in the BA run, implying that BA executes correctly its outputs satisfy validity and consistency.
  - $g = 2$: honest parties input s to the BA instance, which by validity outputs s at all honest parties. Hence, all parties output s from the silent consensus protocol.
  - $g = 3$: by graded agreement, each honest party has $g_i \geq 2$, so each honest party outputs the output of the BA instance. Consistency of the silent consensus protocol hence follows from consistency of the BA subroutine. □

Corollary A.4. *Silent BA (i.e., BA with the silence property for predetermined value s) can be solved in $O(n)$ rounds with bit complexity $O(n^2\kappa)$.*

Proof. We apply Theorem A.3 with the silent 4-graded consensus protocol given by Theorem A.2 and the BA protocol of Momose and Ren [23] using threshold signatures. This protocol runs for $O(n)$ rounds and uses $O(n^2\kappa)$ bits of communication. While not explicitly stated, it is not hard to verify that each party individually sends $O(n)$ messages of size $O(\kappa)$, as on the top level of the recursion it transmits $O(n\kappa)$ messages and participates in exactly one recursive call with at most $\lceil n/2 \rceil$ nodes. □