Near-Optimal Early Stopping with Optimal Communication

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

[1] Yale University
[2] Ruhr University Bochum

Abstract. Minimizing both the round and communication complexity of Byzantine agreement (BA) is a longstanding problem. In the synchronous setting, early stopping protocols aim to terminate in number of rounds that is proportional to the actual number $f \leq t$ of corruptions in an execution, rather than on the worst case threshold t. The lower bound on the round complexity for such protocols is known to be $\min\{f+2, t+1\}$ rounds. Existing protocols, however, achieve either round complexity close to the optimal bound or low communication complexity, but not both simultaneously. Elsheimy, Loss, and Papamanthou (ASIACRYPT 2024) obtain $(1+\epsilon)f + O(1)$ rounds for any $1 > \epsilon > 0$, with $O(n^4 f\kappa)$ communication. In contrast, Lenzen and Sheikholeslami (PODC 2022) achieve asymptotically optimal $O(n \cdot t)$ communication for $t < n/3$, but with a $25f + 13$ number of rounds.
In this work, we present deterministic synchronous Byzantine agreement protocols that achieve new tradeoffs between early stopping round complexity and communication. For $t < n/3$, without cryptographic setup, we obtain $f + O(\sqrt{f \log f})$ termination rounds with optimal $O(n \cdot t)$ communication (here and throughout, $\sqrt{f}$ denotes $\sqrt{f+1}$ and $\log f$ denotes $\log(f+2)$, so that the bounds are well defined when $f = 0$). Thus, our round complexity has the optimal leading coefficient of one, with a sublinear additive overhead. By allowing slightly higher communication, we reduce this overhead to $O(\sqrt{f})$, obtaining $f + O(\sqrt{f})$ rounds with $O(n \cdot t + f^2 \log f)$ communication. Finally, for $t < n/2$, assuming a public key infrastructure and threshold signatures, we obtain $f + O(\sqrt{f})$ rounds with $O(n \cdot t \cdot \kappa)$ bits of communication, where κ is the signature length.

1 Introduction

Byzantine Agreement (BA) is a fundamental problem, first introduced in the seminal work of Lamport, Shostak, and Pease in the 1980s [LSP82]. In the BA problem, n parties, each starting with a value in $\{0, 1\}$, aim to reach agreement on a single value despite the presence of up to $t < n/2$ Byzantine (malicious) parties. The protocol must satisfy two key properties: *agreement*, which ensures that all honest parties decide on the same value, and *validity*, which guarantees that if all honest parties start with the same value, they must agree on that value. Byzantine agreement is a basic building block in many fault tolerant

distributed computing and in secure multiparty computation, and its cost is measured along two main axes: its *round complexity*, which determines how quickly honest parties can terminate, and its *communication complexity*, which measures the total number of bits exchanged. Both measures have been studied extensively, and their optimal behavior is well understood when considered separately. However, optimizing them simultaneously is a highly nontrivial task, as standard strategies for achieving one of these goals typically conflict with each other.

It is well known that any deterministic synchronous BA protocol tolerating up to t corruptions requires $t + 1$ rounds. When the actual number of corruptions f in an execution is much smaller than t, however, it is natural to ask whether the protocol can terminate correspondingly earlier. This question has motivated a long line of work on *early stopping* Byzantine agreement [PT84,DRS90,BGP92b,BG93], where the running time depends on the actual number $f \leq t$ of corruptions rather than only on the worst case threshold t. Dolev, Reischuk, and Strong [DRS90] showed that the optimal early stopping round complexity is $\min\{f + 2, t + 1\}$. This bound was first matched with optimal resilience $n > 3t$ by Berman, Garay, and Perry [BGP92b] (albeit at the cost of exponential communication). Much later, Abraham and Dolev [AD15] obtained a protocol with optimal early stopping and optimal resilience using only polynomial communication complexity. By comparison, in the authenticated setting with $t < n/2$ malicious corruptions, no optimal solution with polynomial communication has yet been proposed. The earliest attempt was by Perry and Toueg in 1984 [PT84], which achieved a suboptimal round complexity of $2 \cdot \min\{f+2, t+1\}$. More recently, Elsheimy, Loss, and Papamanthou [ELP24] reduced the leading constant in the round complexity, achieving $(1+\epsilon)f + O(1)$ rounds for $1 > \epsilon > 0$. And then, Choi, Choi, Liu, and Zikas [CCLZ26] improved the bound to $\min\{f + 2, t + 1\}$ rounds, albeit with exponential communication.

The other side of the problem is communication complexity. Without cryptographic setup, Dolev and Reischuk [DR85] and later Hadzilacos and Halpern [HH93], showed that every deterministic BA protocol tolerating t corruptions requires $\Omega(n \cdot t)$ communication. With threshold signatures and $n = \Theta(t)$, Spiegelman [Spi21] showed a lower bound of $\Omega(ft + t)$ words for deterministic synchronous BA and gave a matching protocol whose communication adapts to f, but whose termination takes $O(n)$ rounds. These results characterize the communication complexity without simultaneously optimizing the round complexity. When requiring optimal early stopping as well, the communication cost becomes worse: Dolev and Lenzen [DL13] show that every deterministic Byzantine consensus protocol that decides within $f + 2$ rounds must send $\Omega(nt + t^2 f)$ messages. Thus, the optimal $\min\{f + 2, t + 1\}$ round complexity cannot in general be combined with optimal $O(nt)$ communication. This raises a natural question: *how much additional round complexity beyond the optimal bound is necessary in order to obtain communication close to $O(nt)$?*

Lenzen and Sheikholeslami [LS22] showed that relaxing the round complexity to $O(f + 1)$ is sufficient. For $t < n/3$, their recursive Phase King protocol

communicates $O(n(t+1))$ bits while terminating in $25f + 13$ rounds. In the other direction, Elsheimy, Loss, and Papamanthou [ELP24] recently reduced this multiplicative gap in the authenticated setting, to $(1+\epsilon)f + O(1)$, with $O(fn^4)\kappa$ communication. The known results therefore leave the following gap:

> *Can deterministic Byzantine agreement terminate within $f+o(f)$ rounds while achieving asymptotically optimal communication?*

We answer this question affirmatively in the unauthenticated model, while narrowing the gap in the authenticated model. Specifically, we obtain the following results:

1. For $t < n/3$, we give an unauthenticated protocol that terminates within $f + O(\sqrt{f})$ rounds and communicates $O(n \cdot t + f^2 \log(f))$ bits.
2. For $t < n/3$, by slightly increasing the additive round overhead to $O(\sqrt{(f)\log(f)})$, we obtain an unauthenticated protocol with $O(n \cdot t)$ bits of communication, which is asymptotically optimal.
3. For $t < n/2$, in the authenticated setting with a PKI and threshold signatures, we obtain a protocol that terminates within $f + O(\sqrt{f})$ rounds and communicates $O(n \cdot t \cdot \kappa)$ bits. Achieving $f + o(f)$ rounds together with asymptotically optimal communication in this setting remains open.

1.1 Technical Overview

From Phase King to King Committees. Our starting point is the classical Phase King paradigm [BGP92a]. A Phase King protocol proceeds through a sequence of phases, each combining graded consensus with a designated king step. Intuitively, graded consensus detects when the honest parties are already in agreement, while an honest king causes them to agree when they are not. More precisely, the grade determines whether a party keeps its current value or adopts the value proposed by the king. The guarantees of graded consensus ensure that, when the king is honest, the king's value is consistent with any value that an honest party keeps. As a result, all honest parties end the phase with the same value and terminate in the following phase. Thus, if a phase fails to make progress, its king must be corrupted. Since every phase takes only a constant number of rounds, charging each unsuccessful phase to its corrupted king gives the classical $O(f+1)$ early stopping bound.

Committee Escalation. Our protocols follow this Phase King structure, but replace the single king by a *king committee*. In phase k, the king committee C^k contains ck parties, where $c = 3$ in the setting without cryptographic setup and $c = 2$ in the authenticated setting. We call a committee *good* if fewer than a $1/c$ fraction of its members are corrupted, and *bad* otherwise. The members of the committee consequently run Byzantine agreement and use its output in place of the king's proposal. If the committee is good, its Byzantine agreement instance guarantees that all honest committee members output the same value

(agreement property) and, if they all enter with the same value v, that this value is output (validity property). The good committee can therefore play the role of an honest king: its output is consistent with any value that an honest party may be forced to hold onto after graded consensus, and all honest parties leave the phase with a common value. As in classical Phase King, they then terminate in the following phase. Consequently, if agreement has not been reached by the end of a phase, its king committee must be bad.

The key difference from classical Phase King is how much corruption is required for a phase to fail. A corrupted king costs the adversary only one corruption, regardless of the phase. We instead choose disjoint king committees whose sizes increase with the phase number. Therefore, for phase k to fail, C^k must contain at least k corruptions. Since the committees are disjoint, if the execution passes through the first K phases without encountering a good king committee, then these phases account for at least $1 + 2 + \cdots + K = \Theta(K^2)$ distinct corruptions. Thus, with only f corruptions, a good king committee must be reached within $O(\sqrt{f})$ phases. We refer to this progressively increasing the size of the king committee as *committee escalation*.

Sequential Escalation. The simplest realization of this idea is the sequential escalation protocol (Section 3), which we describe next in more detail. At the beginning of phase k, all n parties invoke a graded consensus instance GC_k on their current inputs. Graded consensus provides a weak form of agreement together with a local measure of confidence: each party outputs a value and a grade in $\{0, 1, 2, 3\}$, honest parties with nonzero grade output the same value, and the grades of any two honest parties differ by at most one. The grade therefore determines how strongly a party is confident about its output value. A party obtaining grade 3 sets its output value and participates in the next graded consensus instance before terminating.

Suppose now that phase k does not already terminate through graded consensus. The committee C^k runs a Byzantine agreement protocol on the values obtained through graded consensus and multicasts its output to all parties. The graded consensus grade determines how each party uses this committee output. A party with grade 0 or 1 is not sufficiently committed to its output value from graded consensus and therefore adopts a value received from at least $|C^k|/c$ committee members, if one exists, using a fixed tie breaking rule. A party with grade 2, on the other hand, keeps its output value and does not update it. The graded consensus guarantees make these two update rules compatible whenever C^k is good. Let w denote the value sent by the honest members of a good committee C^k. If no honest party obtains grade 2, then every honest party adopts w. Otherwise, suppose some honest party obtains grade 2 on value v. Since honest grades differ by at most one, every honest party has nonzero grade, and thus, outputs the same value v. In particular, every honest member of C^k enters the Byzantine agreement instance with input v. By validity, the committee must therefore decide $w = v$. Hence, whether an honest party keeps its graded consensus output or adopts the committee value, all honest parties end the phase with the same value, and decide by the end of the next phase.

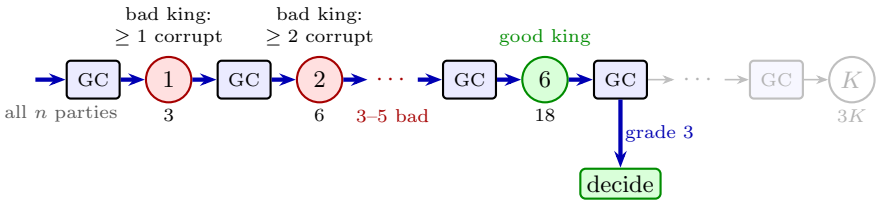

Fig. 1. Sequential committee escalation for $c = 3$. Each phase runs graded consensus (GC) among all n parties, followed by BA in king committee k, which contains $3k$ parties. The number inside each circle is the committee index k, and the number below it is its size. A red king committee is bad, while a green one is good. The blue arrows show the execution. King committees 1 through 5 are bad, so the execution continues. King committee 6, of size 18, is good, so all honest parties enter the following GC with the same value and obtain grade 3. They set their output variable to this value, participate in one additional GC invocation, and then output the value and terminate.

The round complexity follows from charging the cost of each unsuccessful phase to the corruptions required to make its committee bad. The agreement instance in phase k runs among $|C^k| = ck$ parties and tolerates up to $k-1$ corruptions. Since the underlying Byzantine agreement protocol requires $t' + O(1)$ rounds to tolerate t' corruptions, the instance in phase k takes $k + O(1)$ rounds. If phase k fails to make progress because C^k is bad, then C^k contains at least k actual corruptions. We charge the $k + O(1)$ rounds of that phase to these corruptions. Since the committees are disjoint, the same corruption is never charged in two different phases, and the size dependent cost over all bad phases is therefore at most f. Moreover, by the escalation argument, only $O(\sqrt{f})$ bad committees can be encountered. The constant round graded consensus overhead in each phase, together with the first good phase, contributes an additional $O(\sqrt{f})$ rounds. The resulting round complexity is therefore $f + O(\sqrt{f})$.

The sequential construction therefore gives the desired round complexity, but it does not yet provide the communication complexity we are after. Interestingly, the bottleneck is not the Byzantine agreement instances run by the committees. The committees encountered before termination have sizes $c, 2c, \ldots, O(\sqrt{f})$, so even with cubic communication agreement protocols, their total cost is only $O(f^2\kappa)$ authenticated bits, or $O(f^2 \log(f))$ bits when using the protocol of [KM13], which requires no cryptographic setup. Instead, most of the communication is spent on the graded consensus instances surrounding these committee calls. Although the committee itself changes from one phase to the next, graded consensus is still run among all n parties in every phase. Each such invocation costs $O(n^2)$ communication, and there may be $O(\sqrt{f})$ phases before a good committee is reached. This gives $O(n^2\sqrt{f})$ communication, which is precisely the extra factor we would like to avoid.

Recursive Phase King. To remove this repeated cost, we turn to the recursive Phase King framework used by Lenzen and Sheikholeslami [LS22]. Their construction avoids repeating quadratic communication among all n parties in every

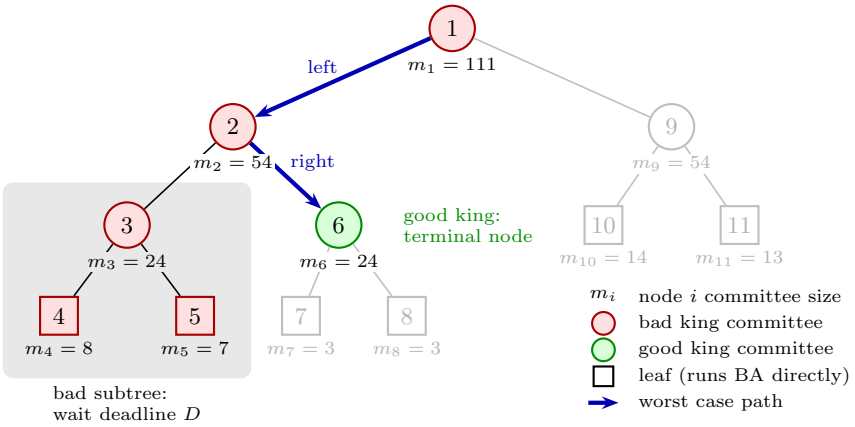

Fig. 2. Recursion tree for $c = 3$ and $n = 111$. The number inside each node is its preorder index i, and m_i denotes the total number of parties assigned to that node. At each internal node k, the king committee contains $3k$ parties; the remaining parties are divided between the left and right children, rounding up for the left child and down for the right child. Squares are leaves, whose entire committees run BA directly. The colors and highlighted path have the same meaning as in Fig. 1.

phase. Rather than executing every phase globally, the parties are organized recursively. The initial invocation contains all n parties and forms the root of the recursion. Each node consists of a committee C participating in one recursive invocation. The parties in C are partitioned into three disjoint subcommittees: a king committee C^0, a left committee C^1, and a right committee C^2. The king committee is used for the Byzantine agreement instance local to that node, while C^1 and C^2 become the two recursive children. The same decomposition is then applied within each recursive child. That is, when the protocol is invoked on C^1, the parties in C^1 are again divided into its own king committee and two smaller recursive committees, and similarly for C^2. This continues only while the current committee contains enough parties to form the prescribed king committee and still leave enough parties for two nonempty recursive children. Once the committee becomes too small for such a split (becomes a leaf), we stop the recursion and simply run Byzantine agreement among all parties in that committee.

The communication advantage comes from this recursive partition. A quadratic cost primitive executed at the root costs $O(n^2)$, but the corresponding primitives in the two children run on committees of size roughly $n/2$, their children on committees of size roughly $n/4$, and so on. Since the groups at the same level are disjoint, the sum of the quadratic costs decreases geometrically down the tree. Consequently, quadratic communication incurred locally at each node can be summed over the entire recursion tree without paying an additional factor equal to the number of Phase King phases. Lenzen and Sheikholeslami use this recursive organization to obtain $O(n \cdot t)$ communication while keeping the $O(f)$ early stopping round complexity of the phase king framework.

There is, however, an important complication when recursion is combined with early stopping. The running time of a recursive call depends on the number of corruptions in the corresponding committee, and is therefore not known to the calling invocation in advance. Waiting for the worst case termination bound of every recursive call would lose early stopping ($O(t)$ per invocation), whereas proceeding as soon as the recursive call terminates creates a synchronization problem: different honest parties may return from the recursive call in different rounds. Lenzen and Sheikholeslami handle this using a synchronization protocol called *pulse*. Once the recursive consensus instance on a committee K produces an output v, the honest members of K broadcast v to all parties. Then a party proceeds only after collecting $n - t$ votes for the same value. The pulse guarantees that if all honest parties vote for the same value, then they all output that value, and that if one honest party outputs in round r, then every honest party outputs by round $r + 1$. Moreover, every pulse output must have been an honest party's input. Thus, recursive calls may exhibit staggered termination, but honest parties return within one round of one another. Lenzen and Sheikholeslami handle this one round difference using a compiler that simulates each round of the underlying protocol by two communication rounds. This constant factor overhead is acceptable for preserving an $O(f)$ round complexity, but it is too costly for our goal of $f + o(f)$ rounds. We therefore need to combine recursive Phase King with committee escalation without paying this multiplicative synchronization overhead.

$f + o(f)$ Rounds via Recursive Committee Escalation. The simulation used by Lenzen and Sheikholeslami is sufficient for preserving an $O(f)$ round complexity, but it is too expensive for our purpose of achieving $f + o(f)$ round complexity. We therefore combine the committee escalation from the sequential construction with the recursive Phase King framework above. We keep the same recursion tree, index its nodes according to their preorder traversal and let the king committee at node k contain ck parties. In this way, the successive king committees play the role of the increasingly large committees in sequential escalation: if the execution moves past node k, then either its king committee is bad and contains at least k corruptions, or the node lies in a bad subtree. The main difficulty is to preserve the leading coefficient of one despite the synchronization issue caused by early returns without relying on the round simulation compiler.

Here, we handle the one round skew by confining it to the graded consensus steps that separate the king committee and the recursive calls. We therefore let graded consensus absorb the skew, so that the expensive step, the recursive call, can still be started by all honest parties in the same round. Concretely, the execution at a node begins with ordinary graded consensus, after which the king committee runs Byzantine agreement. As in the sequential construction, parties that obtain a sufficiently high grade can continue with their graded consensus value, whereas parties with a lower grade use the king committee output. These values are combined through a pulse. The pulse ensures that once one honest party is ready to proceed, every other honest party becomes ready within one additional round. Starting the recursive call immediately from this pulse output

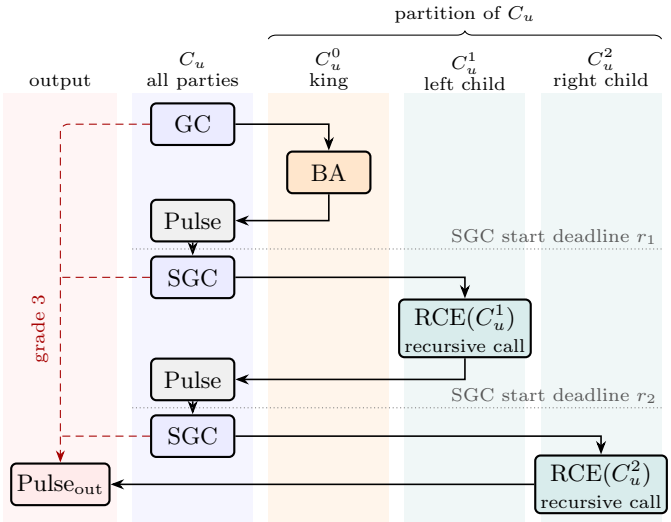

Fig. 3. Execution of one internal node u. The committee C_u is partitioned into a king committee C_u^0 and two child committees C_u^1 and C_u^2. Time flows downward, and each step is placed in the column of the committee that runs it. Each teal box runs the entire protocol recursively on the child committee. A party starts SGC as soon as it gets an output from the preceding pulse, and otherwise at the fixed deadline r_1 or r_2 with its current value. The deadline r_2 includes the full deadline of the left subtree, so a bad left child cannot delay the node indefinitely. A party obtaining grade 3 from GC or SGC inputs its value directly to $\text{Pulse}_{\text{out}}$ and continues executing the protocol until the round following the output from $\text{Pulse}_{\text{out}}$.

would therefore cause different honest parties to invoke the recursive protocol one round apart. Instead, the pulse output is first used as input to another graded consensus instance. Parties that obtain the pulse output earlier start this instance earlier, while any party that has not obtained an output by the predetermined beginning of the next phase starts at that point with its current value. Hence honest parties may start this graded consensus instance in two consecutive rounds.

We therefore use *staggered graded consensus*, which preserves the same graded agreement and graded validity guarantees under such staggered starts. It is obtained by applying the round simulation compiler only to graded consensus, so that each logical communication round is simulated over a constant number of physical rounds and messages sent by parties whose starting times differ by one round are still processed as part of the same logical round. Since graded consensus itself takes only a constant number of rounds, this compilation incurs only constant overhead. Once staggered graded consensus has completed, the left recursive call begins at a fixed round known in advance. Thus, although the preceding graded consensus instance may have started one round apart, the recursive call itself begins at the same time. When the left recursive call returns,

its output plays the same role as the king committee output did before: parties combine the value returned by the child with their current graded consensus value through another pulse, followed by another staggered graded consensus instance. The right recursive call then begins at another fixed round after this staggered instance has completed, and so on.

The recursive calls are then used in the same early stopping manner as in Lenzen and Sheikholeslami. Consider a node whose full committee is good; in particular, this holds for the root node as $n > c \cdot t$. We then examine its three subcommittees in order. If the king committee of the node is good, its Byzantine agreement instance produces a common output that is consistent with the graded consensus values, and the subsequent pulse and staggered graded consensus steps cause the current invocation to terminate with a common output. If the king committee is bad, the protocol proceeds to the left child. Members of the left child that finish the recursive call send their output to the parties in the parent invocation, and parties input the value received to the pulse. If the left child is good, its honest members return the same value, and this leads the parent to terminate early through the pulse and staggered graded consensus steps, without waiting for the full deadline of the left subtree. The deadline is needed only when this does not happen (when the node is bad): once it expires, the protocol stops waiting for the left child and starts the right child.

For the round complexity analysis, it is useful to view the execution as following a single path through the recursion tree. We analyze the execution starting from the root, which contains all n parties and satisfies $f < n/c$. At every node on the path, the parties of the current invocation are partitioned into the king committee, the left child, and the right child. If the king committee is good, the protocol terminates at that node. Otherwise, we first consider the left child. If the left child satisfies the resilience condition, the path continues recursively through it. If the left child is also bad, then the right child must satisfy the resilience condition: since the three sets partition the current committee, they cannot all contain at least a $1/c$ fraction of corruptions while the current committee as a whole contains fewer than a $1/c$ fraction. The path therefore continues through the right child. In this case, the protocol may have waited for the entire deadline of the bad left subtree before doing so. The skipped subtree is bad, so its deadline is bounded by its corruptions plus a constant overhead for each node it contains. The path ends when it reaches either a good king committee, in which case the invocation terminates locally, or a leaf, where BA is run directly.

The round complexity follows by charging the work along this path to corruptions. At every nonterminal node of index k, the king committee is bad and its Byzantine agreement instance takes $k + O(1)$ rounds. Since a bad king committee of size ck contains at least k corruptions, the k nonconstant rounds can be charged to them. A right transition incurs one additional cost: the full deadline of the left subtree. If this subtree contains m parties, its deadline is at most m/c plus a constant overhead for each node in the subtree. Since the subtree is bad, it contains at least m/c corruptions, and hence the nonconstant part of this deadline can be charged to corruptions. The king committees on the path and

the left subtrees are disjoint, so every corruption is charged at most once. Consequently, the total nonconstant work is at most f rounds. It remains to bound the additive overhead from the constant round steps at each node. Let K be the index of the terminal node. As in the sequential construction, reaching a large K requires many corruptions. The only difference is that moving to a right child skips the entire left subtree. In that case, the skipped subtree is bad. Accounting for the bad king committees and skipped bad subtrees gives $K^2 = O(f)$. Hence $K = O(\sqrt{f})$, and the constant overhead over all relevant nodes contributes only $O(\sqrt{f})$ rounds. Thus, the total round complexity is $f + O(\sqrt{f})$.

As for the communication complexity. At a node containing m parties, the graded consensus, pulse, and local message steps require $O(m^2)$ bits in the setting without cryptographic setup and $O(m^2\kappa)$ bits in the authenticated setting. Since the two recursive children have roughly half the size of their parent, these costs sum geometrically to $O(n^2)$ and $O(n^2\kappa)$ bitsalso , respectively. When $n > ct$, we instead run the construction on $ct + 1$ parties. These parties disseminate their output to all parties, and every party outputs the value received from $t+1$ of them. This adds one round, and reduces the local communication to $O(nt)$ and $O(nt\kappa)$, respectively. It remains to account for the BA instances run by the king committees and leaves. These instances follow preorder, and at most one additional leaf BA can begin while the terminal output propagates. Thus at most $K + 1 = O(\sqrt{f})$ BA instances, each involving $O(\sqrt{f})$ parties. In the authenticated setting, their total communication is $O(f^2\kappa)$, which is absorbed by $O(n \cdot t \cdot \kappa)$. Without cryptographic setup, they contribute $O(f^2 \log(f))$ bits. We therefore obtain $f + O(\sqrt{f})$ rounds with $O(n \cdot t \cdot \kappa)$ bits of communication in the authenticated setting, and $O(n \cdot t + f^2 \log(f))$ bits in the other setting.

Slower Escalation for Optimal Communication. To achieve optimal communication, we show how we get rid of the remaining $f^2 \log(f)$ term in the perfectly secure construction. This term comes from the BA instances run by the king committees. We remove this term by slowing the rate at which these committees grow. A bad king committee at index k then contains $\Omega(k/\log(k))$ corruptions. If the path through the recursion tree ends at index K, charging its transitions to the bad king committees and skipped bad subtrees gives $K^2/\log(K) = O(f)$, and hence $K = O(\sqrt{f \log(f)})$. The nonconstant round cost is bounded by the corruptions as before, while the constant local overhead and the final BA instance contribute $O(K)$ rounds. This gives a total round complexity of $f + O(\sqrt{f \log(f)})$. This reduces the cost of the committee agreement instances. At most $K + 1$ such instances are started, each run among $O(K/\log(K))$ parties. Since the perfectly secure Byzantine agreement protocol costs $O(m^3 \log(m + 1))$ bits on m parties, their total communication is $O\left(K\left(\frac{K}{\log(K)}\right)^3 \log(K)\right) = O(K^4/\log^2(K)) = O(f^2)$. This is absorbed by the $O(n \cdot t)$ communication of the recursive construction on $3t + 1$ parties and dissemination to all parties, giving $O(n \cdot t)$ total communication and therefore asymptotically optimal communication.

1.2 Related Work

Early stopping without setup. Garay and Moses [GM98] give a binary Byzantine agreement protocol with optimal resilience $n > 3t$, polynomial communication, and a proved early stopping bound of $\min\{f + 5, t + 1\}$ rounds. Their construction is an intermediate step between the exponential communication protocol of Berman, Garay, and Perry [BGP92b] and the round optimal polynomial communication protocol of Abraham and Dolev [AD15].

Table 1. Synchronous early stopping protocols. Communication is measured in bits for binary inputs, and κ denotes the size of a signature or cryptographic word. The Det. column indicates whether the protocol is deterministic, and the Exp. column indicates whether it terminates in an expected constant number of rounds. All protocols assume synchronous authenticated point to point channels. The protocol of [EKL25] assumes that messages sent before corruption cannot subsequently be removed. The setup column uses PKI to denote a public key infrastructure, TS for threshold signatures, VRF for verifiable random function, and CC for common coin. The common coin can be implemented using VRFs following a standard approach, as in [EKL25], without additional asymptotic cost. The protocols of [EKL25,BLL26] may fail with negligible probability. Neither [AD15] nor [CFLZ26] states a concrete communication bound. In [AD15], every party relays a value for each party to all parties in the second round, giving at least $\Omega(n^3)$ bits, while their analysis only bounds each information gathering tree by $O(n^7)$ times the size of the corrupt tree (the part the adversary keeps unresolved). In the setup-free protocol of [CFLZ26], the leader election alone runs n^2 VSS instances per phase [KK06], giving $\Omega(n^6 f \log n)$ bits in the worst case. The values α and ϵ are arbitrary constants between 0 and 1.

Reference	Resilience	Det.	Exp.	Rounds	Communication	Setup
[LS22]	$n > 3t$	✓	×	$O(f)$	$O(nt)$	None
This work	$n > 3t$	✓	×	$f + O(\sqrt{f \log(f)})$	$O(nt)$	None
This work	$n > 3t$	✓	×	$f + O(\sqrt{f})$	$O(nt + f^2 \log(f))$	None
[AD15]	$n > 3t$	✓	×	$\min\{f + 2, t + 1\}$	Polynomial	None
[ELP24]	$n > 2t$	✓	×	$(1 + \epsilon)f + O(1)$	$O(n^4 f\kappa)$	PKI
This work	$n > 2t$	✓	×	$f + O(\sqrt{f})$	$O(nt\kappa)$	PKI, TS
[CCLZ26]	$n > 2t$	✓	×	$\min\{f + 2, t + 1\}$	Exponential	PKI
[LN24]	$t < n$	✓	×	$O(\min\{f^2, t + 1\})$	$O(n^4(\kappa + \min\{f^2, t\}))$	PKI
[DKLZ25]	$t < (1 - \alpha)n$	✓	×	$O(f + 1)$	$O(n^4(\kappa + \min\{f^2, t\}))$	PKI
[BLL26]	$n > 2t$	×	×	$O(f)$	$O(nf\kappa)$	PKI
[EKL25]	$t < (1-\alpha)n/2$	×	✓	$(1 + \epsilon)f + O(1)$	$O(n\lambda^3\kappa)$	PKI, VRF
[ELP24]	$n > 2t$	×	✓	$(1 + \epsilon)f + O(1)$	$O(n^4 f\kappa)$	PKI, CC
[CFLZ26]	$n > 3t$	×	✓	$O(f)$	Polynomial	None

Early stopping with an honest majority. For $n > 2t$, Perry and Toueg [PT84] give an authenticated early stopping protocol with $2\min\{f +$

$2, t+1\}$ rounds and polynomial communication. Elsheimy, Loss, and Papamanthou [ELP24] obtain $(1+\epsilon)f + O(1)$ termination rounds, with $O((f+1)n^4\kappa)$ bits of communication. Choi, Choi, Liu, and Zikas [CCLZ26] improve the bound to $\min\{f+2, t+1\}$ rounds, albeit with exponential communication.

Constantinescu et al. [CDPW26] achieve $O(n \cdot f)$ communication and ensure that all honest parties decide within $O(f+1)$ rounds when $t < n/2$ and $t = \Omega(n)$. This is not early stopping: their protocol permits honest parties to continue sending messages after deciding and terminate within $O(n)$ rounds, and their proof does not establish that an honest party can safely terminate when it decides. This differs both from protocols such as that of Elsheimy et al. [ETP24], which may require $O(t)$ rounds even to decide despite having only $O(n \cdot f)$ communication, and from early stopping protocols such as ours, in which honest parties terminate within $O(f)$ rounds. These results therefore separate three notions: adaptive communication, early decision, and early termination. Dufay, Paramonov, and Wattenhofer [DPW26] extend early decision to L bit inputs with optimal authenticated resilience $t < n/2$. Their synchronous constructions achieve $O(n(L+f\kappa))$ bits and $O(f+\log n)$ decision rounds, or $\widetilde{O}(n\kappa+t(L+f\kappa))$ bits and $O(f)$ decision rounds, where $\widetilde{O}$ suppresses polylogarithmic factors. They still incur $O(n)$ termination rounds.

Elsheimy, Loss, and Papamanthou [ELP24] also give a randomized authenticated protocol with constant expected round complexity and worst case early stopping bound of $(1+\epsilon)f + O(1)$ rounds. Elsheimy, Kamp, and Loss [EKL25] combine constant expected round complexity with an early stopping bound of $(1+\epsilon)f + O_\epsilon(1)$ rounds and $O(n\lambda^3\kappa)$ communicated bits with overwhelming probability, where λ is a statistical security parameter. Their construction tolerates $t < (1-\alpha)n/2$ corruptions for a fixed $\alpha > 0$, uses a committee election functionality, and assumes that messages sent before a party's corruption cannot subsequently be removed. Blum, Lenzen, and Loss [BLL26] obtain $O(f)$ termination rounds and $O(nf\kappa)$ bits with failure probability $2^{-\kappa}$ under a PKI and optimal resilience $t < n/2$. Their protocol tolerates a strongly adaptive adversary that can remove or alter the current round messages of parties corrupted in that round. Ciampi, Forghani, Lu, and Zikas [CFLZ26] give a compiler that lifts a broad class of expected constant round BA protocols, including those of Feldman and Micali and of Katz and Koo, to additionally terminate within $O(f)$ rounds in the worst case. This yields protocols for $t < n/3$ without setup and, assuming a PKI, for $t < n/2$. They extend this to parallel broadcast and obtain MPC with expected constant rounds and $O(f)$ worst case rounds. Their focus is on combining expected constant round complexity with an $O(f)$ worst case bound, whereas we consider deterministic protocols and aim for the optimal leading coefficient of one in the worst case round complexity, $f + o(f)$, together with optimal communication.

Early stopping with a dishonest majority. Loss and Nielsen [LN24] give a deterministic broadcast protocol under a plain PKI that tolerates any $t < n$ corruptions and terminates in $O(\min\{f^2, t+1\})$ rounds with polynomial communication. Their polarizer construction provides transferable evidence of missing

messages despite the absence of an honest majority. Deligios, Klasovitá, and Liu-Zhang [DKLZ25] improve the termination bound to $O(f)$ whenever $t < (1-\epsilon)n$ for a fixed $\epsilon > 0$. Their construction uses a plain PKI, tolerates strongly adaptive corruptions, and matches the communication complexity of Loss and Nielsen.

Communication efficient agreement. Momose and Ren [MR21] give authenticated BA protocols with $O(n^2\kappa)$ bits of communication, using either threshold signatures for $t < n/2$ or a standard PKI for $t \leq (1/2-\epsilon)n$.

Without a PKI, Dolev and Reischuk [DR85] show an $\Omega(nt)$ message lower bound for deterministic BA, even in a failure free execution; Hadzilacos and Halpern [HH91] give a similar lower bound using related connectivity based proof. Dolev and Reischuk also show that, when every message carries a non-compressible signature, some failure free execution transmits $\Omega(nt)$ signature occurrences, yielding $\Omega(nt\kappa)$ bits if each signature is sent explicitly.

These signature lower bounds do not apply to stronger setup that compresses many signatures into constant size certificates. In the threshold signature model, Spiegelman [Spi21] proves a tight $\Theta(ft+t)$ word bound, which is $\Theta(nf)$ when $n = \Theta(t)$. Related adaptive communication results include $O(nf)$ words for Byzantine broadcast [CKS22], $O(nf)$ words for deterministic synchronous binary BA using threshold signatures and SNARKs [ETP24], and $O(nL+nf\kappa)$ bits for strong multivalued validated agreement [CDG$^+$24].

1.3 Version History

A preliminary manuscript by a subset of the present authors gives an authenticated BA protocol with $f + O(\sqrt{f+1})$ rounds and polynomial communication, together with an EIG based broadcast protocol with $\min\{f+3, t+1\}$ rounds and exponential communication [ELP25]. The present paper subsumes its polynomial communication construction by reducing the authenticated communication to $O(n(t+1)\kappa)$ bits and obtaining optimal communication without cryptographic setup.

1.4 Use of AI

All technical ideas and results in this work are our own. Large language models, specifically ChatGPT and Claude, were used extensively to assist with revising and editing the manuscript, including the presentation of related works table and figures. We take full responsibility for the final content, including any errors.

1.5 Paper Organization

Section 3 presents sequential escalation, Section 4 presents recursive escalation, and Section 4.1 analyzes its security and complexity. Appendix A gives the pulse protocols, Appendix B gives the GC and SGC instantiations, Appendix C gives the authenticated BA protocol, and Appendix D gives the detailed proofs for recursive escalation.

2 Preliminaries

Network and Setup Assumptions. We assume a fully connected network consisting of n parties $\{P_1, \ldots, P_n\} = \mathcal{P}$, where each pair of parties shares an authenticated communication channel. We consider a *synchronous network model*, meaning all parties have access to a synchronized clock and there exists a known upper bound Δ on the delivery time of messages between honest parties. Consequently, protocols can proceed in synchronized rounds of length Δ, such that any message sent by an honest party at the start of a round is guaranteed to be delivered to all honest parties by the end of that round.

Adaptive Byzantine Corruptions. We assume that an adversary may adaptively corrupt up to $t < n/c$ parties for any positive integer $c \geq 2$. Corrupted parties are Byzantine and may deviate arbitrarily from the protocol. The adversary is able to observe the messages sent by honest parties and, in case it chooses to corrupt the sender, prevent them from being delivered to their honest recipient. For a fixed execution, F denotes the set of all parties corrupted at any point during that execution. In correctness statements, an honest party is a party outside F.

Definition 1 (Good and Bad Party Sets and Instances). *Whenever our protocols target a corruption bound $t < n/c$, we say that a set of parties S is good with respect to a corruption set F if $|S \cap F| < |S|/c$. Otherwise S is bad. Similarly, we refer to a protocol instance executed by a good party set as good, and otherwise bad. When F is clear from context, we simply say that the party set or instance is good or bad.*

Authentication. For $c = 3$, we assume only the authenticated point to point channels described above and require no cryptographic setup. For those of our results where $c = 2$, we assume the existence of a public key infrastructure (PKI) supporting a digital signature scheme with efficient signing and verification algorithms, denoted Sign and Verify, respectively. Each party P_i holds a public key pk_i known to all other parties, and a corresponding private key sk_i known only to P_i. A party P_i can generate a signature on message m as $\mathsf{sig}(m) := \mathsf{Sign}(\mathsf{sk}_i, m)$, which can be verified using $\mathsf{Verify}(\mathsf{pk}_i, \mathsf{sig}(m), m)$. We say a signature $\mathsf{sig}(m)$ is *valid* if $\mathsf{Verify}(\mathsf{pk}_i, \mathsf{sig}(m), m) = 1$. For notational convenience, we use the shorthand $\langle m \rangle_i$ to denote the signed message tuple $(m, \mathsf{sig}(m))$ produced by party P_i. Similarly, we will use $\langle m \rangle_i^\tau$ and Σ_m^τ to denote, respectively, a partial signature from party P_i on m and a threshold signature resulting from combining partial signatures on m from τ distinct parties. For authenticated pulse and graded consensus, the setup provides a unique $(n - t)$ threshold signature scheme for the n participating parties. Partial and combined threshold signatures have size $O(\kappa)$ bits. Finally, We model signatures as ideal primitives with perfect unforgeability.

Protocol Conventions. We specify each protocol for n participating parties and corruption threshold t, except for protocols such as recursive committee escalation that explicitly distinguish multiple committees. Communication counts

bits sent by parties while they are honest, including before an adaptive corruption. The BA communication bounds below are sums of local sending bounds that hold whenever the configured corruption threshold is respected. Before a transmission would exceed its bound, a party aborts that BA instance without producing an output. This preserves executions within the threshold and makes the same communication bounds apply when the threshold is exceeded. The graded consensus and pulse constructions have the same communication bounds even when their corruption thresholds are exceeded, by their explicit sending rules instructing each party to send only a constant number of bits. A party decides when it irrevocably fixes its BA value, outputs when it makes the result available, and terminates when it ceases participating, having already output. All BA round bounds measure termination. When a party terminates an invocation, it cancels every unfinished subprotocol instance that it invoked, recursively.

Byzantine Agreement. The main problem we are targeting is Byzantine agreement (BA) [LSP82].

Definition 2 (Byzantine Agreement). *Let Π be protocol executed among parties $P_1, ..., P_n$, where each party P_i holds an input $v_i \in \{0,1\}$ and outputs a value $y_i \in \{0,1\}$ before or upon terminating. A protocol Π achieves Byzantine Agreement, if the following properties hold whenever at most t parties are corrupted.*

- ***Validity:*** *If every honest party P_i inputs $v_i = v$, then all honest parties output $y_i = v$;*
- ***Agreement:*** *All honest parties output the same value v.*
- ***T-Termination:*** *If all honest parties invoke Π in round r, then all honest parties terminate by the end of round $r + T - 1$.*

In addition to the early stopping protocols we construct, we will be using the BA protocol by Kowalski and Mostéfaoui [KM13] for $n > 3t$ and a folklore BA based on Dolev and Strong [DS83b] for $n > 2t$ which we describe in Section C. The two fixed round BA instantiations output and terminate in their final round.

Graded Consensus. We use graded consensus, which takes an input from each party and outputs a value alongside one of 4 grades: 0, 1, 2, or 3. The grade indicates the degree of certainty in common knowledge.

Definition 3 (Graded Consensus). *Let Π be a protocol executed by parties $P_1, ..., P_n$, where each party P_i inputs $v_i \in \{0,1\}$ and outputs a pair $(y_i, g_i) \in \big(\{0,1\} \times \{0, \ldots, G\}\big) \cup \{(\perp, 0)\}$ before or upon terminating. A protocol Π achieves Graded Consensus if the following properties hold whenever at most t parties are corrupted.*

- ***Graded Validity:*** *If all honest parties P_i have the same input value $v_i = v$ then all honest parties output $y_i = v$ and $g_i = G$.*

- ***Graded Agreement:*** *Let P_i and P_j denote honest parties that output y_i, g_i and y_j, g_j, respectively. Then (1) $|g_i - g_j| \leq 1$ and (2) $g_i, g_j \geq 1$ implies that $y_i = y_j$.*
- ***T-Termination:*** *If all honest parties invoke Π in round r, then all honest parties terminate by the end of round $r + T - 1$.*

In some cases, we need a variant of graded consensus which tolerates that honest parties invoke the protocol one round apart. We call such protocol Staggered Graded Consensus (SGC).

Definition 4 (Staggered Graded Consensus). *Let Π be a protocol executed by parties $P_1, ..., P_n$, where each party P_i inputs $v_i \in \{0, 1\}$ and outputs a pair $(y_i, g_i) \in \big(\{0, 1\} \times \{0, \ldots, G\}\big) \cup \{(\perp, 0)\}$ before or upon terminating. A protocol Π achieves Staggered Graded Consensus if the following properties hold whenever at most t parties are corrupted and, for some round r, every honest party invokes Π in either round r or round $r + 1$.*

- ***Graded Validity:*** *If all honest parties P_i have the same input value $v_i = v$ then all honest parties output $y_i = v$ and $g_i = G$.*
- ***Graded Agreement:*** *Let P_i and P_j denote honest parties that output y_i, g_i and y_j, g_j, respectively. Then (1) $|g_i - g_j| \leq 1$ and (2) $g_i, g_j \geq 1$ implies that $y_i = y_j$.*
- ***Staggered T-Termination:*** *If $r_{\max}$ is the latest honest invocation round, all honest parties terminate by the end of round $r_{\max} + T - 1$.*

We use $G = 3$ in every GC and SGC invocation. The constructions in Section B instantiate GC with the graded consensus protocols of Fitzi, Liu-Zhang, and Loss [FLZL21]: one for $t < n/3$, and another for $t < n/2$. They take 3 and 4 communication rounds and send $O(n^2)$ and $O(n^2\kappa)$ bits, respectively. The explicit compiler we use yields SGC with termination rounds 9 and 12, respectively, and the same asymptotic communication costs.

Synchronization Pulse. In order to guarantee that parties will invoke the staggered graded consensus protocol at most one round apart, we make use of a synchronization primitive that we call a *pulse* protocol. It abstracts techniques commonly used to synchronize without a PKI when $n > 3t$ [ST87] or with a PKI when $n > 2t$ [ADD$^+$19]. A pulse runs from a common starting round $r_{\texttt{start}}$, but each party P_i may receive its input x_i in an arbitrary, possibly late, round r_i. It provides three guarantees. First, if all honest parties input the same value x in round r or earlier, then all honest parties receive output x by round r. This ensures that the protocol can continue once all honest parties are ready. Next, if an honest party P_i gets output x_i in round r, then every honest party P_j gets output x_j no later than in round $r + 1$. Note that this does not imply any agreement properties for the outputs. Finally, it has a validity property which says that an output y_i of an honest party P_i must be equal to an input x_j already input by some honest party P_j. Thus, if all honest inputs supplied before an output agree, then that output preserves their agreement.

Definition 5 (Pulse). *Let Π be a protocol executed by parties $P_1, \ldots, P_n$ from a common starting round $r_{\texttt{start}}$. Each honest party participates from that round, may receive one input x_i at the beginning of a later round, and outputs a value y_i before or upon terminating. A party may output without ever receiving an input. The protocol implements* pulse *if the following hold whenever at most t parties are corrupted and all honest parties participate from round $r_{\texttt{start}}$ until the round mentioned in the respective guarantee.*

- ***Termination:*** *If every honest party has received the same input by the beginning of round r, then every honest party terminates by the end of round r.*
- ***Output Staggering:*** *If some honest party outputs in round r, then every honest party outputs by the end of round $r + 1$.*
- ***Validity:*** *Every honest output in round r equals an input supplied to an honest party by the beginning of round r.*

3 Warmup: Sequential Escalation

We first present *sequential committee escalation*, a simple realization of committee escalation that is secure against $t < n/c$ corruptions for any integer $c > 1$, assuming graded consensus and BA protocols with the same resilience. The protocol proceeds through a sequence of phases with disjoint king committees $(C_1, C_2, \ldots)$, where (C_k) has size (ck). In phase k, all parties first run graded consensus on their current votes, after which the members of (C_k) run BA on the resulting values and send their output to all parties. A party with grade 0 or 1 adopts any value received from at least $|C_k|/c$ members of C_k, if such a value exists, whereas a party with grade at least 2 keeps its graded consensus output. A party obtaining grade 3 additionally decides that value. The purpose of the increasing committee sizes is that a phase can fail to bring the honest parties to a common vote only if its king committee is bad. Since a bad C_k contains at least k corruptions and the committees are disjoint, later unsuccessful phases require progressively more distinct corruptions.

We will use this protocol in two settings. For $c = 3$, we instantiate graded consensus with the three round graded consensus construction of [FLZL21] and byzantine agreement with the protocol of Kowalski and Mostéfaoui [KM13]; this requires no cryptographic setup, and tolerates $t < n/3$. The latter has $O(n^3 \log(n + 1))$ communication complexity. For $c = 2$, we use the four round graded consensus construction of [FLZL21] together with the authenticated BA protocol of Appendix C, tolerating resilience $t < n/2$ under a PKI and threshold signatures. The latter is a folklore adaptation of Dolev–Strong [DS83a]: a value is accepted only if it carries an input certificate of $t + 1$ input signatures, which by honest majority guarantees that some honest party input it, and it is then relayed along chains of forwarding signatures following Dolev-Strong logic. Each party accepts at most two values and outputs v only if v is the single value it accepted (and 0 otherwise). The protocol has the same communication complexity

of [DS83a] of $O(n^3\kappa)$ bits. We first analyze sequential escalation abstractly for general c, and instantiate these bounds at the end of the section.

Sequential Escalation

The parties $P_1, \ldots, P_n$ are partitioned into committees $C^1, \ldots, C^K$, where K is the smallest integer satisfying $c\frac{K(K+1)}{2} \geq n$. For each $k < K$, the committee C^k has size $|C^k| = c \cdot k$, whereas C^K includes the remaining parties giving it size $|C^K| = n - c\frac{K(K-1)}{2} \leq c \cdot K$.
We use δ_Π to denote the number of rounds of a protocol Π. Each instance BA^k is configured to tolerate $q_k := \lceil |C^k|/c \rceil - 1$ corruptions. Let r_1 be the common starting round and define $r_{k+1} := r_k + \delta_{\mathsf{GC}} + \delta_{\mathsf{BA}^k} + 2$ for $k < K$. Thus each phase allows one round for disseminating the committee output and one round for the final local updates before the next GC begins.
Each party P_i holds an input value x_i, and a variable $y_i = \perp$ which defines the output upon termination.
P_i sets its initial vote $\mathsf{vote}_i^1 := x_i$ and proceeds in phases as follows for $k = 1, \ldots, K$:

- **In round r_k:** Each party P_i runs GC^k with input vote_i^k. Let (v_i^k, g_i^k) denote the output.
- **In round $r_k + \delta_{\mathsf{GC}}$:** If P_i has $y_i \neq \perp$, it terminates. Each P_i in C^k runs BA^k with input v_i^k if $v_i^k \neq \perp$, otherwise with input vote_i^k. If it receives an output from BA^k, it sends it to all parties.
- **In round $r_k + \delta_{\mathsf{GC}} + \delta_{\mathsf{BA}^k} + 1$:** Party P_i defines its vote for the next round as follows:
  - If P_i has $g_i^k < 2$ and some value v was received from at least $\frac{|C^k|}{c}$ parties in C^k, it defines $\mathsf{vote}_i^{k+1} := v$.
  - Otherwise, it defines $\mathsf{vote}_i^{k+1} := v_i^k$ if $v_i^k \neq \perp$, and $\mathsf{vote}_i^{k+1} := \mathsf{vote}_i^k$ otherwise.
  - If $g_i^k = 3$, P_i sets $y_i := v_i^k$.

If P_i has not terminated after iteration K, it sets $y_i := \mathsf{vote}_i^{K+1}$ and terminates.

Fig. 4. Sequential committee escalation for general corruption thresholds

3.1 Security Analysis

We first show how a good committee makes all honest proposals agree.

Lemma 1 (Good Phase). *Suppose that fewer than $\frac{|C^k|}{c}$ parties in C^k are corrupted and no honest party decides before phase k. Then all honest parties set vote_i^{k+1} to a common value w and decide w in phase $\min\{k+1, K\}$ or earlier.*

Proof. Since no honest party has decided before phase k, all honest parties participate in GC^k, and hence GC^k satisfies Definition 3.

We first establish that every honest party sets vote_i^{k+1} to a common value. Because fewer than $\frac{1}{c}|C^k|$ parties in C^k are corrupted, BA^k satisfies agreement and validity. All honest parties in C^k therefore obtain a common output w from BA^k and send w to every party. The honest parties in C^k number at least $|C^k| - \frac{1}{c}|C^k| = \frac{c-1}{c}|C^k| \geq \frac{1}{c}|C^k|$, so every honest party receives w from at least $\frac{|C^k|}{c}$ parties in C^k. Conversely, since only parties in C^k send BA^k outputs and every honest one sends w, any value $w' \neq w$ is sent by at most the fewer than $\frac{1}{c}|C^k|$ corrupted parties. Hence w is the *unique* value that any honest party receives from at least $\frac{|C^k|}{c}$ parties in C^k. Consider how each honest party P_i sets vote_i^{k+1}:

- If $g_i^k < 2$, then P_i sets $\mathsf{vote}_i^{k+1} := v$ for a value v it received from at least $\frac{|C^k|}{c}$ parties in C^k. By the uniqueness above, $v = w$.
- If $g_i^k \geq 2$, then P_i sets $\mathsf{vote}_i^{k+1} := v_i^k$. Since honest grades differ by at most one, every honest party then has grade at least 1, so by graded agreement all honest parties output the same value v from GC^k. In particular, every honest party in C^k inputs v to BA^k, so by validity of BA^k its output is $w = v$. Thus P_i sets $\mathsf{vote}_i^{k+1} = v = w$.

In both cases P_i sets $\mathsf{vote}_i^{k+1} := w$, so all honest parties adopt the common value w. Any honest party that decides in phase k therefore decides w. If $k = K$, all honest parties set $y_i := \mathsf{vote}_i^{K+1} = w$ and terminate at the end of the phase. If $k < K$, all honest parties participate in GC^{k+1} with the common input w, including those that decided in phase k. Graded validity makes every honest output from GC^{k+1} equal to $(w, 3)$, so every honest party that has not already decided sets $y_i := w$ at the end of phase $k + 1$.

We now show that the protocol satisfies agreement. In fact we show a slightly stronger statement which will be useful in the following analysis.

Lemma 2 (Agreement). *If an honest party P_i sets $y_i := v \neq \perp$ in phase k, then every honest party sets $y_j := v$ by phase $k + 1$ or earlier.*

Proof. Assume WLOG that k is the first phase in which some honest party P_i decides $y_i = v \neq \perp$. First suppose that $k < K$, so P_i got output $(v, 3)$ from GC^k.

We first want to argue that GC^k and GC^{k+1} satisfy Definition 3. Note that an honest party P_j terminates either after phase K, or in some phase k' where P_j has $y_j \neq \perp$ and is done executing $\mathsf{GC}^{k'}$. Thus, since k is the first phase in which an honest party decides, all honest parties participate in GC^k and GC^{k+1} and we can invoke Definition 3. Since GC^k satisfies Definition 3, we know that all honest parties have output $(v, 3)$ or $(v, 2)$. Thus every honest party P_j will set $\mathsf{vote}_j^{k+1} := v$. Some honest parties (those with $g_j^k = 3$) decide v in phase k but still participate in GC^{k+1}, while the others enter phase $k + 1$ with $y_j = \perp$. In phase $k + 1$, all honest parties participate in GC^{k+1} with input v, and thus all honest parties get output $(v, 3)$ from GC^{k+1}. Any honest party P_j that did not already set $y_j = v$ sets $y_j = v$ at the end of phase $k + 1$.

Now suppose that $k = K$. Since the committees partition all parties and fewer than n/c parties are corrupted, at least one committee C^h is good: otherwise their total number of corruptions would be at least $\sum_{j=1}^{K} |C^j|/c = n/c$. No honest party decides before phase h, so Lemma 1 implies that all honest parties decide a common value in phase K or earlier. In particular, the assignments $y_j := \mathsf{vote}_j^{K+1}$ at the end of the final phase all give the same value v.

Lemma 3 (Unanimity). *Let k be a phase in which all honest parties participate in GC^k. If every honest party inputs the same value v to GC^k, then every honest party that has not already decided sets $y_i := v$ at the end of phase k and decides v.*

Proof. Since all honest parties participate in GC^k and give it the common input v, graded validity (Definition 3) guarantees that every honest party obtains output $(v, 3)$; that is, $v_i^k = v$ and $g_i^k = 3$. As $g_i^k = 3$, every honest party that has not already decided sets $y_i := v_i^k = v$ at the end of phase k and hence decides v.

Lemma 4 (Validity). *If all honest parties have the same input v, then all honest parties output v.*

Proof. Assume all honest parties have the same input v. Then all honest parties input v to GC^1. Since GC^1 satisfies graded validity (Definition 3) all honest parties get output $(v, 3)$ from GC^1 and hence decide v.

Lemma 5 (Termination). *Assume Fig. 4 uses a GC protocol with $O(1)$ round complexity and a BA^j with $q_j + O(1)$ round complexity, each tolerating fewer than a $1/c$ fraction of corruptions among its participants, for the fixed integer $c \geq 2$. If at most $f \leq t < n/c$ parties are corrupted, then all honest parties terminate in $f + O(\sqrt{f+1})$ rounds.*

Proof. Let C^k be the first good committee, which exists because the committees partition all parties and $f < n/c$. If some honest party decides before phase k, Lemma 2 makes all honest parties decide by phase k. Otherwise, Lemma 1 makes all honest parties decide by phase $\min\{k+1, K\}$. Each party terminates after the next GC invocation following its decision, or at the end of phase K if that comes first. Hence all honest parties terminate after at most $k+2$ GC invocations and $k+1$ committee BA invocations. Every committee C^j with $j < k$ is bad and has size cj, so it contains at least j corrupted parties. Since these committees are disjoint, $f \geq \sum_{j=1}^{k-1} j = \frac{k(k-1)}{2}$, and therefore $k = O(\sqrt{f+1})$. Each committee BA tolerates $q_j = \lceil |C^j|/c \rceil - 1 \leq j - 1$ corruptions, including for the final committee. The total number of BA rounds is consequently at most $\sum_{j=1}^{\min\{k+1,K\}} \big(q_j + O(1)\big) \leq \frac{k(k+1)}{2} + O(k+1) \leq f + O(k+1)$. The GC invocations and the remaining local steps add $O(k+1)$ rounds, giving $f + O(\sqrt{f})$ rounds in total.

Theorem 1 (Sequential escalation instantiations). *For every execution with $f \leq t$ adaptive corruptions, sequential escalation protocol gives the following deterministic binary Byzantine agreement protocols:*

1. *for $t < n/2$, with a PKI and unique threshold signatures, $f + O(\sqrt{f})$ rounds and $O(n^2\sqrt{f}\kappa)$ bits;*
2. *for $t < n/3$, with perfect security and without cryptographic setup, $f + O(\sqrt{f})$ rounds and $O(n^2\sqrt{f})$ bits.*

Proof. For $c = 2$ and $c = 3$, use the instantiations described at the beginning of this section. In both cases, graded consensus takes $O(1)$ rounds, while a committee BA instance configured to tolerate q corruptions takes $q + O(1)$ rounds. The preceding security and termination analysis therefore gives validity, agreement, and $f + O(\sqrt{f})$ rounds in both settings.

There are $O(\sqrt{f})$ graded consensus invocations among all n parties, contributing $O(n^2\sqrt{f}\kappa)$ and $O(n^2\sqrt{f})$ bits, respectively. The committee Byzantine agreement instances have sizes $O(j)$ for $1 \leq j \leq O(\sqrt{f})$, so their total communication is $O(f^2\kappa)$ and $O(f^2 \log(f))$ bits, respectively. The committee outputs sent to all parties contribute another $O(nf)$ bits. Since $f < n$ and $\log(f) = O(\sqrt{f})$, these costs are absorbed by the graded consensus communication, proving both bounds.

4 Recursive Escalation

We now present *recursive committee escalation*, which reduces the communication cost of the sequential construction while preserving its near-optimal round complexity. The communication bottleneck in sequential escalation is graded consensus: every phase invokes GC among all n parties, and repeating this quadratic communication over $O(\sqrt{f})$ phases gives $O(n^2\sqrt{f})$ bits, with an additional κ in the authenticated setting. We reduce this cost by running the surrounding graded consensus steps on smaller committees.

We do so by combining committee escalation with recursive phase king [MR21,LS22]. Each recursive invocation is associated with a committee C. At each internal node, the parties are partitioned into a king committee and two smaller committees on which the protocol can recurse. Graded consensus now runs only among the parties of the current invocation, and the two child committees each contain at most half as many parties as their parent. Without cryptographic setup, these quadratic communication costs sum to $O(n^2)$ over the entire tree, or $O(nt)$ after restricting participation to $O(t)$ parties and disseminating their output. The authenticated costs carry an additional factor of κ accounting for (threshold) signatures. For the basic instantiation, an internal node at preorder index k assigns ck parties to its king committee, so the king committees grow in the order in which the recursion is scheduled.

Combining early stopping and recursive phase king requires special care. As each node in the tree acts as an early stopping protocol, its output round is controlled by the adversary, who can force honest parties to "stagger" and obtain output in different rounds [Nie03]. Lenzen and Sheikholeslami [LS22] handle this stagger using a pulse and a round simulation compiler, obtaining $O(f)$ rounds with $O(nt)$ communication. Applying such a simulation to the entire protocol would multiply the rounds charged to faults and lose our $f + O(\sqrt{f})$ guarantee.

Staggered graded consensus.(Appendix B) We instead apply round simulation compiler only to graded consensus, which already has constant round complexity. The recursive protocol will guarantee that honest parties invoke a GC instance at most one round apart. We therefore need a variant of graded consensus that preserves the guarantees of Definition 3 under such staggered invocations. More generally, let Π be an R-round synchronous protocol, and suppose that every honest party invokes Π in one of two consecutive rounds. We simulate each logical round of Π over three physical rounds. Messages are tagged with their logical round, and each party accepts messages for a logical round throughout a collection window determined by its own invocation time. Because honest parties start at most one round apart, every message sent by an honest party in logical round j falls within the round j collection window of every honest recipient. Moreover, all such collection windows close before any honest party sends a message for logical round $j+1$. The messages accepted in each logical round therefore define a valid synchronous execution of Π.

Lemma 6. *Suppose Π is an R-round synchronous protocol and all honest parties invoke its staggered simulation in rounds r or $r+1$. The simulation preserves the correctness properties of Π, including against adaptive corruptions. If $r_{\max}$ is the latest honest invocation round, every honest party outputs by the end of round $r_{\max} + 3R - 1$, and honest output rounds differ by at most one. If Π sends at most B bits in at most M messages, the simulation sends at most $B + O(M \log(R+1))$ bits in at most M messages.*

Applying Lemma 6 to the graded consensus protocols used in this paper gives deterministic grade 3 SGC with 9 termination rounds and $O(n^2)$ communication for $t < n/3$, and 12 termination rounds and $O(n^2\kappa)$ communication for $t < n/2$ under the authenticated setup.

We use this construction only at the boundaries between the king committee and the recursive calls. Whenever a committee or recursive invocation produces an output, a pulse ensures that honest parties become ready at most one round apart. They then invoke SGC with these staggered starting times. The following recursive call begins at a fixed round after the SGC instance, and therefore all honest parties start the recursive call simultaneously. Thus, the round simulation overhead is confined to constant round graded consensus rather than being applied to the recursive computation itself. This timing also allows the protocol to achieve early stopping. If GC or SGC returns grade 3, its value is immediately used to trigger the termination pulse of the current invocation. Otherwise, the protocol continues according to fixed deadlines. If a king committee or recursive child does not produce an output that leads to early termination, the protocol waits only until the corresponding deadline before moving on. As we show below, the nonconstant part of such a wait can be charged to corruptions in a bad king committee or subtree, while the remaining local work contributes only constant overhead per visited node. With king committees of size ck, recursive committee escalation achieves $f + O(\sqrt{f})$ rounds with $O(nt\kappa)$ communication in the authenticated setting and $O(nt + f^2 \log(f))$ communication without cryptographic

setup. We also consider a slower escalation schedule. Using king committees of size $\Theta(k/\log(k+1))$ reduces the communication without cryptographic setup to $O(nt)$ bits, at the cost of increasing the round complexity to $f + O(\sqrt{f \log(f)})$.

Recursion tree and deadlines. We describe both constructions using a public committee size schedule $s : \mathbb{N}_{>} 0 \to \mathbb{N}_{>} 0$. We consider nondecreasing schedules satisfying $s(k) \leq k$ and $s(2k) = O(s(k))$. At the node with preorder index k, the protocol RCE_s assigns $c \cdot s(k)$ parties to the king committee. The basic construction uses $s(k) = k$. The root contains $(P_1, \ldots, P_n)$ and has preorder index 1. At an internal node containing m parties with index k, the first $cs(k)$ parties form the king committee and the remaining parties form two child committees of sizes m_L and m_R with indices $k+1$ and k_R, respectively. For an internal node, set $m_L = \left\lceil \frac{m - c \cdot s(k)}{2} \right\rceil$, $m_R = \left\lfloor \frac{m - c \cdot s(k)}{2} \right\rfloor$, and $k_R = k + 1 + N_s(m_L, k+1)$. We define $N_s : \mathbb{N}_{>0} \times \mathbb{N}_{>0} \to \mathbb{N}$ as

$$N_s(m,k) = \begin{cases} 1 & \text{if } 0 < m \leq c(s(k)+1), \\ 1 + N_s(m_L, k+1) + N_s(m_R, k_R) & \text{otherwise.} \end{cases}$$

The first case covers that the node is a leaf, while the second sums the number of nodes in the left and right subtree. At an internal node, $m - cs(k) \geq c+1 \geq 3$, so $m_L, m_R \geq 1$ and both recursive calls are nonempty. A recursive child may be bad and need not terminate. Waiting for it to terminate will kill our early stopping goals. We therefore assign every subtree a fixed deadline after which its parent stops waiting for it. For every good recursive invocation, this deadline is an upper bound on its running time. The deadline pessimistically allocates the full recursive deadline to both children. Let $T_{\mathsf{BA}}(m)$ be a fixed upper bound on the number of rounds occupied by the BA protocol among m parties, and let d_{loc} be a fixed upper bound on the local delay at an internal node excluding its BA instance and recursive calls. We assume that $T_{\mathsf{BA}}(m) \leq \lceil m/c \rceil + d_{\mathsf{BA}}$ for a constant d_{BA}. We define $D_s : \mathbb{N}_{>0} \times \mathbb{N}_{>0} \to \mathbb{N}$ by

$$D_s(m,k) = \begin{cases} T_{\mathsf{BA}}(m) & \text{if } 0 < m \leq c(s(k)+1), \\ T_{\mathsf{BA}}(c \cdot s(k)) + D_s(m_L, k+1) & \\ \qquad + D_s(m_R, k_R) + d_{\mathrm{loc}} & \text{otherwise.} \end{cases}$$

Committee primitives We instantiate every subprotocol locally for the committee C executing the current recursive invocation. Each instance is configured to tolerate $\lceil |C|/c \rceil - 1$ corruptions, and an instruction to send to all parties within that instance means to send only to the parties in C. We use the grade 3 GC and SGC constructions of Section B, and tag all messages with the current recursive invocation and phase. The remaining primitives depend on the resilience setting. For $c = 2$, we use the authenticated pulse of Fig. 6 with $n = |C|$ and $t = \lceil |C|/2 \rceil - 1$. The setup provides a unique threshold signature scheme for each committee C, with threshold $\lfloor |C|/2 \rfloor + 1$, which is also used by the authenticated GC and SGC instances. For $c = 3$, we use the pulse of Fig. 7 with $n = |C|$ and $t = \lceil |C|/3 \rceil - 1$.

Timing For a protocol Π, let δ_Π denote a fixed upper bound on the duration of a protocol Π. We take $(\delta_{\mathsf{GC}}, \delta_{\mathsf{SGC}}) = (3, 9)$ without cryptographic setup and $(4, 12)$ in the authenticated setting. An invocation that begins at the start of round r and has duration δ terminates, and hence has output, by the end of round $r+\delta-1$. For a committee C, set $\delta_{\mathsf{BA}(C)} := T_{\mathsf{BA}}(|C|)$, and let $d_{\mathrm{loc}} := \delta_{\mathsf{GC}}+2\delta_{\mathsf{SGC}}+7$. A pulse has no unconditional duration. However, if all honest inputs are available at the beginning of round r, every honest party receives its pulse output by the end of round r and may use it at the beginning of round $r + 1$.

Output timing in recursive calls. The parent uses a recursive child's output as soon as it is available, while the child continues its prescribed participation until termination. This composition relies on every honest member of a good child invocation having output by its first honest termination, provided no ancestor cancels it first.

Recursive Escalation $\mathsf{RCE}_s(C = (P_1, \ldots, P_m), k, r_0)$

If $m \leq c(s(k)+1)$, run $\mathsf{BA}(C)$ with input x_i and output its result as y_i and terminate in its final round; otherwise, continue below.
Set $x_i^0 := x_i$, $x_i^1 = x_i^2 = y_i := \perp$, and $b_i^{\text{out}} := \perp$. All three pulses start in round r_0. Each "once" instruction is executed at most once; each pulse takes only the first input supplied by P_i; pulse outputs precede simultaneous timeouts; and value ties are resolved as $0 < 1$. Upon output z from $\mathsf{Pulse}^{\text{out}}$ in round r, party P_i sets $y_i := z$ and $b_i^{\text{out}} := \top$ and outputs z from the current invocation. It continues to participate for one additional round and terminates the invocation at the end of round $r+1$.
King phase.

1. **In round** r_0**:** P_i runs GC^0 with input x_i^0 and lets (v_i^0, g_i^0) be the output.
2. **In round** $r_0 + \delta_{\mathsf{GC}}$**:** If $g_i^0 = 3$ and $b_i^{\text{out}} = \perp$, P_i inputs v_i^0 to $\mathsf{Pulse}^{\text{out}}$ and sets $b_i^{\text{out}} := \top$. If $g_i^0 \geq 2$, P_i inputs v_i^0 to Pulse^0. Let $C^0 := (P_1, \ldots, P_{c \cdot s(k)})$. Each party P_i in C^0 runs $\mathsf{BA}(C^0)$ with input v_i^0 if $v_i^0 \neq \perp$, otherwise with input x_i^0. Upon obtaining an output w, it sends w to all parties.
3. **Once** w **is received from** $\lceil |C^0|/c \rceil$ **parties in** C^0**:** If $g_i^0 < 2$, P_i inputs w to Pulse^0.

Left phase. Let $r_1 = r_0 + \delta_{\mathsf{GC}} + \delta_{\mathsf{BA}(C^0)} + 2$ be its fixed starting round.

4. **Once** Pulse^0 **outputs** w**:** P_i sets $x_i^1 := w$ if $x_i^1 = \perp$.
5. **In round** r_1**:** If $x_i^1 = \perp$, set $x_i^1 := x_i^0$.
6. **Once** $x_i^1 \neq \perp$**:** P_i runs SGC^1 with input x_i^1.
7. **Once** SGC^1 **outputs** (v_i^1, g_i^1)**:** If $g_i^1 = 3$ and $b_i^{\text{out}} = \perp$, input v_i^1 to $\mathsf{Pulse}^{\text{out}}$ and set $b_i^{\text{out}} := \top$. If $g_i^1 \geq 2$, input v_i^1 to Pulse^1.
8. **In round** $r_1 + \delta_{\mathsf{SGC}}$**:** Let $\ell = \left\lceil \frac{m - c \cdot s(k)}{2} \right\rceil$ and $C^1 = (P_{c \cdot s(k)+1}, \ldots, P_{c \cdot s(k)+\ell})$. Each party P_i in C^1 invokes $\mathsf{RCE}_s(C^1, k+1, r_1 + \delta_{\mathsf{SGC}})$ with input v_i^1 if $v_i^1 \neq \perp$, otherwise with input x_i^1. Upon obtaining an output w from the recursive call, it sends w to all parties.
9. **Once** w **is received from** $\lceil |C^1|/c \rceil$ **parties in** C^1**:** If $g_i^1 < 2$, P_i inputs w to Pulse^1.

Right phase. Let $r_2 = r_1 + \delta_{\mathsf{SGC}} + D_s(\ell, k+1) + 2$ be its fixed starting round.

10. **Once** Pulse^1 **outputs** w**:** P_i sets $x_i^2 := w$ if $x_i^2 = \perp$.
11. **In round** r_2**:** If $x_i^2 = \perp$, set $x_i^2 := x_i^1$.
12. **Once** $x_i^2 \neq \perp$**:** P_i runs SGC^2 with input x_i^2.
13. **Once** SGC^2 **outputs** (v_i^2, g_i^2)**:** If $g_i^2 = 3$ and $b_i^{\text{out}} = \perp$, input v_i^2 to $\mathsf{Pulse}^{\text{out}}$ and set $b_i^{\text{out}} := \top$.
14. **In round** $r_2 + \delta_{\mathsf{SGC}}$**:** Let $C^2 = (P_{c \cdot s(k)+\ell+1}, \ldots, P_m)$. Each party P_i in C^2 invokes $\mathsf{RCE}_s(C^2,\ k+1+N_s(\ell, k+1), r_2 + \delta_{\mathsf{SGC}})$ with input v_i^2 if $v_i^2 \neq \perp$, otherwise with input x_i^2. Upon obtaining an output w from the recursive call, it sends w to all parties.
15. **Once** w **is received from** $\lceil |C^2|/c \rceil$ **parties in** C^2**:** If $b_i^{\text{out}} = \perp$, P_i inputs w to $\mathsf{Pulse}^{\text{out}}$ and sets $b_i^{\text{out}} := \top$.

Fig. 5. Recursive committee escalation with committee size schedule s, executed at one node of the preorder tree.

4.1 Security Analysis

We first establish the local properties used to prove agreement and termination. The detailed proofs are deferred to Section D.1. We then combine these local properties by following a worst case path through the recursion tree. This path identifies the recursive invocation whose output determines the outcome of the execution and will also be used to bound the round complexity.

Local properties. Throughout, consider an invocation of $\mathsf{RCE}_s(C, k)$ such that $|C \cap F| < |C|/c$ and every honest member of C enters the invocation. A good king committee plays the role of an honest king in Phase King. If graded consensus gives some honest party grade at least 2 on a value v, then graded agreement gives every honest member of the king committee value v with grade at least 1, so each enters BA with v. Otherwise, the common BA output of the good king committee determines the value carried forward.

Lemma 7 (Good King Committee). *If C^0 is good, then all honest outputs from the current invocation agree. Every honest party outputs by the end of round $r_1 + \delta_{\mathsf{SGC}}$ and terminates by the end of the following round.*

The reason is the same as in Phase King: the honest members of a good king committee agree on the BA output, and any value obtained by an honest party with grade at least 2 must agree with that output. The following pulse and SGC steps therefore bring all honest parties to the same value. We next relate the output of a good recursive child to the output of the preceding staggered graded consensus.

Lemma 8 (Compatibility with a child output). *Let $j \in \{1, 2\}$ and suppose that C^j is good. If an honest member of C^j outputs w from the recursive invocation executed by C^j and an honest party outputs (v, g) from SGC^j with $g \geq 2$, then $w = v$.*

We now show that if a good child completes with output w by its deadline, then the current invocation also completes with output w.

Lemma 9 (Good Child Committee). *Let $j \in \{1, 2\}$. Suppose that C^j is good and every honest member of C^j outputs w from its recursive invocation by the end of round $r_j + \delta_{\mathsf{SGC}} + D_s(|C^j|, k_j) - 1$, where $k_1 = k + 1$ and $k_2 = k_R$. Suppose that no honest party inputs to $\mathsf{Pulse}^{\mathtt{out}}$ in Line 2 or, when $j = 2$, in Line 7. Then every honest output from the current invocation equals w, and every honest party outputs and terminates.*

To see why, all honest outputs of the good child equal w. By Lemma 8, any high grade output of the corresponding SGC also has value w. Hence all honest inputs that can reach the next pulse agree on w, and the subsequent pulse and SGC steps preserve this value and complete the current invocation. We now use these local properties to identify the recursive call that determines the worst case execution. If the king committee is good, the current invocation completes

by Lemma 7. Otherwise, the king committee is bad. If the left child is good, we continue the analysis with the left recursive invocation. If the left child is also bad, then the right child must be good because the three subcommittees partition a good committee. Repeating this argument leads to either a leaf or a node with a good king committee, giving the following worst case path.

Worst case path. Fix the corruption set F, with $|F| = f < n/c$. Define the worst case path $u_1, \ldots, u_r$ starting from the root. At a node u_i, the path ends if u_i is a leaf or its king committee $C^0_{u_i}$ is good. Otherwise, if the left child $C^1_{u_i}$ is good, the path continues to that child; if both $C^0_{u_i}$ and $C^1_{u_i}$ are bad, it continues to the right child. We call u_r the *terminal node*.

Lemma 10. *Every node on the worst case path is good.*

Proof. The root is good because $f < n/c$, and the path enters a left child only when that child is good. For a right transition, both the king committee and left child of a good node u are bad. Since C^0_u, C^1_u, C^2_u partition C_u, the right child must be good; otherwise $|C_u \cap F| \geq (|C^0_u| + |C^1_u| + |C^2_u|)/c = |C_u|/c$, a contradiction.

So every invocation on the path satisfies the resilience condition. The terminal node either runs BA at a good leaf or has a good king committee, and Lemmas 7 to 9 ensure its common output is preserved through the preceding invocations.

Round complexity. For a node u, write $n_u := |C_u|$ and let k_u be its preorder index. The king committee of an internal node contains ck_u parties, so its BA instance takes $k_u + O(1)$ rounds. All other local steps take only a constant number of rounds. We will use two simple bounds on a subtree rooted at u (proved in Lemmas 25 and 26). First, it contains $N_u = O(\sqrt{n_u/c + 1})$ nodes: the king committees of its internal nodes are disjoint and have distinct positive indices, so a subtree with I internal nodes already contains $\Omega(cI^2)$ parties. Second, its full deadline satisfies $D_u \leq n_u/c + O(N_u)$, since the size dependent BA costs over all king committees and leaves sum to at most n_u/c, while every node contributes only constant additional overhead. We now bound the cost of the worst case path by charging each transition to corrupted parties.

Lemma 11 (Corruption cost of a left transition). *Suppose the path transitions from u_i to its left child. Then the king committee $C^0_{u_i}$ is bad and contains at least k_{u_i} corrupted parties.*

Lemma 12 (Corruption cost of a right transition). *Suppose the path transitions from u_i to its right child u_{i+1}. Then $C^0_{u_i} \cup C^1_{u_i}$ contains $\Omega(k^2_{u_{i+1}} - k^2_{u_i})$ corrupted parties.*

For a right transition, this charge includes both the bad king committee and the skipped bad left subtree. In particular, if the skipped subtree contains I internal nodes, the combined size of its disjoint king committees implies that the bad subtree contains $\Omega(Ik_{u_i} + I^2)$ corruptions, while the transition increases the preorder index by $2I + 2$.

Lemma 13 (Preorder index of the terminal node). *The terminal node u_r satisfies $k_{u_r} = O(\sqrt{f})$.*

Proof. Let $k_i := k_{u_i}$. For a left transition, $k_{i+1} = k_i + 1$, so $k_{i+1}^2 - k_i^2 = O(k_i)$, which is paid for by Lemma 11. For a right transition, Lemma 12 pays for $k_{i+1}^2 - k_i^2$. The charged king committees and skipped subtrees are pairwise disjoint, so summing over the path gives $k_r^2 - k_1^2 = O(f)$. Since $k_1 = 1$, the claim follows.

Theorem 2 (Early stopping round complexity). *If an execution contains $f < n/c$ corrupted parties, all honest parties terminate within $f + O(\sqrt{f})$ rounds.*

Proof. The worst case path argument shows that the running time is bounded by the work at the nodes on the path(see Lemma 28). At every nonterminal node u_i, the king BA costs at most $k_{u_i} + O(1)$ rounds, while its bad king committee contains at least k_{u_i} corruptions. If a bad left subtree v is skipped, its deadline satisfies $D_v \le n_v/c + O(N_v)$ (Lemma 26); since v is bad, $n_v/c \le |C_v \cap F|$. The king committees on the path and the skipped left subtrees are disjoint, so all size dependent costs sum to at most f. Let $K := k_{u_r} = O(\sqrt{f})$ by Lemma 13. The path contains at most K nodes, and the skipped subtrees together contain at most K nodes, since all of these nodes occur before the terminal node in preorder. Hence their constant overhead is $O(K)$. The terminal BA also takes $O(K)$ rounds. Therefore the total running time is $f + O(K) = f + O(\sqrt{f})$.

Finally, we argue validity and agreement. Validity is immediate from the first graded consensus step: if all honest parties start with the same value v, they all obtain $(v, 3)$ and input v to $\mathsf{Pulse}^{\mathsf{out}}$, so pulse validity makes every honest output equal to v (see Lemma 23). Agreement follows by the backward induction along the worst case path described above: the terminal invocation has a common output, and each preceding invocation preserves this value (see Lemma 28).

Communication complexity. At a node u, the SGC, pulse, and local dissemination steps communicate $O(n_u^2)$ bits without cryptography and $O(n_u^2\kappa)$ bits in the authenticated setting. Since the two children satisfy $|C_u^1|^2 + |C_u^2|^2 \le n_u^2/2$, these costs sum to $O(n^2)$ and $O(n^2\kappa)$ over the recursion tree. By Lemma 13, the terminal index is $O(\sqrt{f})$. Moreover, every BA instance started before termination lies in the preorder prefix ending at the terminal node, except possibly one additional leaf started while the terminal output propagates. Thus only $O(\sqrt{f})$ BA instances begin, each among $O(\sqrt{f})$ parties. In the authenticated setting, the BA protocol has communication $O(m^3\kappa)$ on m parties, so the BA instances contribute $O(f^2\kappa)$ bits. Without cryptographic setup, the BA contributes $O(f^2 \log(f))$ bits. Restricting the recursive protocol to $O(t)$ parties when necessary gives total communication $O(nt\kappa)$ in the authenticated setting and $O(nt + f^2 \log(f))$ without cryptographic setup.

For optimal communication, set $s(k) := \lceil k/\log_2(k+1) \rceil$. Since $\sum_{k=1}^{K} s(k) = \Theta(K^2/\log(K+1))$, the same charging argument gives $K = O(\sqrt{f \log f})$. At most $O(K)$ BA instances begin, each among $O(s(K))$ parties, for total communication $O(Ks(K)^3 \log(s(K)+1)) = O(K^4/\log^2(K+1)) = O(f^2)$, which is absorbed by the $O(nt)$ communication of the recursive construction.

References

[AD15] Ittai Abraham and Danny Dolev. Byzantine agreement with optimal early stopping, optimal resilience and polynomial complexity. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, page 605–614, New York, NY, USA, 2015. Association for Computing Machinery.

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

[BG93] Piotr Berman and Juan A Garay. Cloture votes: n/4-resilient distributed consensus in t+ 1 rounds. *Mathematical systems theory*, 26:3–19, 1993.

[BGP92a] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Bit optimal distributed consensus. *Computer Science*, pages 313–321, 1992.

[BGP92b] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Optimal early stopping in distributed consensus. In Adrian Segall and Shmuel Zaks, editors, *Distributed Algorithms*, pages 221–237, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[BLL26] Erica Blum, Christoph Lenzen, and Julian Loss. Early-stopping consensus with adaptive bit complexity. Cryptology ePrint Archive, Paper 2026/951, 2026.

[CCLZ26] Wonseok Choi, Wookjin Choi, Xiangyu Liu, and Vassilis Zikas. On the round complexity of early stopping. Cryptology ePrint Archive, Paper 2026/1878, 2026.

[CDG+24] Pierre Civit, Muhammad Ayaz Dzulfikar, Seth Gilbert, Rachid Guerraoui, Jovan Komatovic, and Manuel Vidigueira. DARE to agree: Byzantine agreement with optimal resilience and adaptive communication. In Ran Gelles, Dennis Olivetti, and Petr Kuznetsov, editors, *43rd ACM PODC*, pages 145–156. ACM, June 2024.

[CDPW26] Andrei Constantinescu, Marc Dufay, Anton Paramonov, and Roger Wattenhofer. From few to many faults: Optimal adaptive byzantine agreement. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC 2026)*, pages 177–187. Association for Computing Machinery, 2026.

[CFLZ26] Michele Ciampi, Pouyan Forghani, Yun Lu, and Vassilis Zikas. Randomized early-stopping Byzantine agreement with applications to round-efficient MPC. Cryptology ePrint Archive, Paper 2026/2284, 2026.

[CKS22] Shir Cohen, Idit Keidar, and Alexander Spiegelman. Brief announcement: Make every word count: Adaptive byzantine agreement with fewer words. In Alessia Milani and Philipp Woelfel, editors, *41st ACM PODC*, pages 421–423. ACM, July 2022.

[DKLZ25] Giovanni Deligios, Ivana Klasovita, and Chen-Da Liu-Zhang. Asymptotically optimal early termination for dishonest majority broadcast. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part V*, volume 15605 of *LNCS*, pages 116–144. Springer, Cham, May 2025.

[DL13] Danny Dolev and Christoph Lenzen. Early-deciding consensus is expensive. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*, PODC '13, page 270–279, New York, NY, USA, 2013. Association for Computing Machinery.

[DPW26] Marc Dufay, Anton Paramonov, and Roger Wattenhofer. Brief announcement: Optimal adaptive multi-valued byzantine agreement. In *40th International Symposium on Distributed Computing (DISC 2026)*, 2026. To appear. Full version: arXiv:2608.17552.

[DR85] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *Journal of the ACM*, 32(1):191–204, 1985.

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

[DS83a] Danny Dolev and H. Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12:656–666, 11 1983.

[DS83b] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.

[EKL25] Fatima Elsheimy, Simon Holmgaard Kamp, and Julian Loss. Early stopping is cheap. Cryptology ePrint Archive, Paper 2025/1107, 2025.

[ELP24] Fatima Elsheimy, Julian Loss, and Charalampos Papamanthou. Early stopping byzantine agreement in $(1+\epsilon)\cdot f$ rounds. In *Advances in Cryptology – ASIACRYPT 2024, Part VI*, volume 15489 of *Lecture Notes in Computer Science*, pages 398–424. Springer, 2024.

[ELP25] Fatima Elsheimy, Julian Loss, and Charalampos Papamanthou. Towards optimal early stopping agreement protocols. Cryptology ePrint Archive, Report 2025/246, 2025.

[ETP24] Fatima Elsheimy, Giorgos Tsimos, and Charalampos Papamanthou. Deterministic byzantine agreement with adaptive $O(n \cdot f)$ communication. In David P. Woodruff, editor, *35th SODA*, pages 1120–1146. ACM-SIAM, January 2024.

[FLZL21] Matthias Fitzi, Chen-Da Liu-Zhang, and Julian Loss. A new way to achieve round-efficient byzantine agreement. In Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen, editors, *40th ACM PODC*, pages 355–362. ACM, July 2021.

[GM98] Juan Garay and Yoram Moses. Fully polynomial byzantine agreement for n ¿ 3t processors in t + 1 rounds. *SIAM Journal on Computing*, 27(1):247–290, 1998.

[HH91] Vassos Hadzilacos and Joseph Y. Halpern. Message-optimal protocols for byzantine agreement (extended abstract). In Luigi Logrippo, editor, *10th ACM PODC*, pages 309–323. ACM, August 1991.

[HH93] Vassos Hadzilacos and Joseph Y. Halpern. Message-optimal protocols for byzantine agreement. *Mathematical Systems Theory*, 26(1):41–102, 1993.

[KK06] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006*, pages 445–462, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[KM13] Dariusz R. Kowalski and Achour Mostéfaoui. Synchronous byzantine agreement with nearly a cubic number of communication bits: synchronous byzantine agreement with nearly a cubic number of communication bits. In Panagiota Fatourou and Gadi Taubenfeld, editors, *32nd ACM PODC*, pages 84–91. ACM, July 2013.

[LN24] Julian Loss and Jesper Buus Nielsen. Early stopping for any number of corruptions. In *Advances in Cryptology – EUROCRYPT 2024: 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26–30, 2024, Proceedings, Part III*, page 457–488, Berlin, Heidelberg, 2024. Springer-Verlag.

[LS22] Christoph Lenzen and Sahar Sheikholeslami. A recursive early-stopping phase king protocol. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC'22, page 60–69, New York, NY, USA, 2022. Association for Computing Machinery.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[MR21] Atsuki Momose and Ling Ren. Optimal communication complexity of authenticated byzantine agreement. In *35th International Symposium on Distributed Computing (DISC 2021)*, volume 209 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

[Nie03] Jesper Buus Nielsen. *On Protocol Security in the Cryptographic Model*. PhD thesis, Department of Computer Science, University of Aarhus, August 2003. BRICS Dissertation Series DS-03-8.

[PT84] Kenneth Perry and Sam Toueg. An authenticated byzantine generals algorithm with early stopping. *Cornell eCommons*, 1984. Online.

[Spi21] Alexander Spiegelman. In search for an optimal authenticated byzantine agreement. In *35th International Symposium on Distributed Computing (DISC 2021)*, volume 209 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 38:1–38:19, 2021.

[ST87] T. K. Srikanth and Sam Toueg. Optimal clock synchronization. *J. ACM*, 34(3):626–645, 1987.

A Pulse

A pulse synchronizes parties that receive their inputs at different times. We give two binary protocols satisfying Definition 5, each executed by n parties $P_1, \ldots, P_n$, of which at most t are corrupted. The authenticated construction tolerates $t < n/2$ and communicates $O(n^2\kappa)$ bits using threshold signatures. The unauthenticated construction tolerates $t < n/3$ and communicates $O(n^2)$ bits using only authenticated channels. In both constructions, every honest output equals an input already supplied to an honest party, and the first honest output is followed by all remaining honest outputs within one round. Different honest parties may output different bits when honest inputs differ.

All honest parties participate from the common starting round $r_{\texttt{start}}$, including before receiving an input. Each party receives at most one input bit, supplied at the beginning of a round, and processes messages at the end of each round. An input triggered message sent at the beginning of round r is delivered by the end of round r. A message sent in response to the messages processed at the end of round r is delivered and processed in round $r + 1$. Such responses are sent before the sender outputs and terminates. A party counts its own messages as received in their delivery round. Every threshold counts distinct parties, and parties ignore messages for other instances, values outside $\{0, 1\}$, and invalid signatures. When both bits satisfy a rule, they are processed in the order $0, 1$. The communication bounds count the messages and payload bits sent by honest parties in a single instance.

A.1 Authenticated Pulse with an Honest Majority

Assume $t < n/2$ and a unique threshold signature setup for the n parties with threshold $n - t$. Each party has a signing share, and both partial and combined signatures have size $O(\kappa)$ bits. On receiving input x, a party broadcasts its partial signature on $(\mathsf{sid}, \texttt{ready}, x)$, where sid identifies the pulse instance. Any $n - t$ valid partial signatures from distinct parties can be combined into a threshold signature on that message. Since at least one of these shares belongs to an honest party, the combined signature certifies that an honest party has received input x.

A party that obtains a combined signature, either by combining shares or by receiving it from another party, forwards it to all parties and outputs its certified value. Verification and forwarding do not require the recipient to have received an input. Thus one honest output makes a combined signature available to every remaining honest party in the following round. Figure 6 gives the protocol.

Authenticated pulse with an honest majority

Each party P_i, $i \in \{1, \ldots, n\}$, holds a signing share for a unique $(n - t)$ threshold signature scheme. All parties can verify partial and combined signatures. All signatures are on $(\mathsf{sid}, \texttt{ready}, x)$, where sid identifies this pulse instance. Each party P_i holds an input variable x_i, initially $\perp$ and set to a bit when it receives input, and an output variable $y_i = \perp$. It records valid partial signatures, indexed by signer and value.

Beginning of round r: If P_i receives input x_i in this round, it computes its partial signature σ_i on $(\mathsf{sid}, \texttt{ready}, x_i)$ and sends $(\texttt{share}, x_i, \sigma_i)$ to all parties.

End of round r: After recording the received messages, if some bit x has either valid partial signatures from $n-t$ distinct parties or a valid combined threshold signature Σ_x, choose the smallest such x, combine its shares into Σ_x if needed, send $(\texttt{ready}, x, \Sigma_x)$ to all parties for delivery in round $r + 1$, set $y_i = x$, and terminate.

Fig. 6. A pulse protocol for n parties with $t < n/2$, using unique $(n - t)$ threshold signatures.

Lemma 14. *The protocol in Fig. 6 satisfies Definition 5 for n parties with at most $t < n/2$ corruptions, assuming a unique $(n - t)$ threshold signature setup.*

Proof. **Validity:** A valid combined signature requires partial signatures from $n - t > t$ distinct parties, including at least one honest party. An honest party produces its partial signature only after receiving the signed bit as its input for this instance. Thus every honest output equals an input already supplied by an honest party.

Output Staggering: Before outputting in round r, an honest party sends its combined signature to all parties. Every honest party that has not yet output receives and verifies this signature by the end of round $r + 1$ and therefore outputs by that time.

Termination: Suppose every honest party receives the same input x by the beginning of round r. If an honest party outputs in a round $s < r$, output staggering guarantees that every honest party outputs by round $s + 1 \leq r$. Otherwise, every honest party sends its partial signature on x by the beginning of round r. The at least $n - t$ honest shares allow every honest party to combine a threshold signature and output by the end of round r. Validity ensures that all these outputs equal x.

Lemma 15 (Communication complexity). *The honest parties in the protocol in Fig. 6 send $O(n^2)$ messages containing $O(n^2\kappa)$ bits in total.*

Proof. Each honest party sends its partial signature to all n parties at most once. It also sends at most one combined threshold signature to all parties, since it terminates immediately afterward. Both types of messages contain a bit and one signature of size $O(\kappa)$. Thus each honest party sends $O(n)$ messages containing $O(n\kappa)$ bits, and all honest parties together send $O(n^2\kappa)$ bits. These send bounds hold even when the number of corruptions exceeds t.

A.2 Unauthenticated Pulse with an Honest Supermajority

Assume $t < n/3$. A party announces its input x by sending $(\texttt{ready}, x)$ to all parties. It also relays this message after receiving it from $t + 1$ distinct parties, since at least one sender must be honest. Tracing such relays backward reaches an honest party that sent the message because it received input x. Each party sends at most one ready message for each bit.

A party outputs x after receiving ready messages for x from $2t + 1$ distinct parties. For the first honest output x, at least $t + 1$ of the supporting senders are honest, so every honest party receives enough messages to relay x. The resulting honest broadcasts provide at least $n - t \geq 2t + 1$ ready messages at every remaining honest party in the next round. To preserve these broadcasts when a party outputs, Fig. 7 executes every enabled relay before its output rule.

Lemma 16. *The protocol in Fig. 7 satisfies Definition 5 for n parties with at most $t < n/3$ corruptions.*

Proof. **Validity:** Every honest output x is supported by $2t + 1$ ready messages, including at least one from an honest sender. Consider an honest party that first sends $(\texttt{ready}, x)$. If it were relaying $t + 1$ received messages, at least one would have been sent earlier by an honest party, contradicting this choice. Hence an honest party received input x before any honest party sent a ready message for x, and therefore before any honest party output x.

Unauthenticated pulse with an honest supermajority

Each party P_i, $i \in \{1, \ldots, n\}$, holds an input variable x_i, initially $\perp$ and set to a bit when it receives input, and an output variable $y_i = \perp$. It also maintains a set S_i of values for which it has already sent a `ready` message, initially $S_i = \emptyset$. It records ready messages by sender and value.

Beginning of round r: If $x_i \neq \perp$ and $x_i \notin S_i$, send $(\texttt{ready}, x_i)$ to all parties and add x_i to S_i.

End of round r: Record the received messages, then execute the following steps in order:

1. For each bit $x \notin S_i$ with ready messages from at least $t+1$ distinct parties, send $(\texttt{ready}, x)$ to all parties for delivery in round $r+1$ and add x to S_i.
2. If some bit x has ready messages from at least $2t+1$ distinct parties, choose the smallest such x, set $y_i = x$, and terminate.

Fig. 7. A pulse protocol for n parties with $t < n/3$, using authenticated channels.

Output Staggering: Let r be the first round in which an honest party outputs, and let x be its output. Among the $2t+1$ ready messages supporting this output, at least $t+1$ were sent by honest parties for delivery in round r or earlier. Every honest party has therefore received those $t+1$ messages by the end of round r. Since no honest party terminated earlier, each either already sent a ready message for x or sends one for delivery in round $r+1$ before executing its output rule. Consequently, every honest party that has not yet output receives at least $n - t \geq 2t+1$ ready messages for x by the end of round $r+1$ and outputs by then.

Termination: Suppose every honest party receives the same input x by the beginning of round r. If an honest party outputs in a round $s < r$, output staggering gives all honest outputs by round $s+1 \leq r$. Otherwise, every honest party sends $(\texttt{ready}, x)$ by the beginning of round r, either upon receiving its input or as an earlier relay. Every honest party therefore receives at least $n - t \geq 2t+1$ such messages by the end of round r and outputs by that time. Validity ensures that all these outputs equal x.

Lemma 17 (Communication complexity). *The honest parties in the protocol in Fig. 7 send $O(n^2)$ messages containing $O(n^2)$ bits in total.*

Proof. For each value $x \in \{0, 1\}$, an honest party sends $(\texttt{ready}, x)$ to all parties at most once. Thus each honest party sends at most two broadcasts, or $O(n)$ point to point messages, and all honest parties together send $O(n^2)$ messages. Since a `ready` message carries only a binary value and a constant size tag, the total communication is $O(n^2)$ bits. These send bounds hold even when the number of corruptions exceeds t.

B Graded Consensus and Staggered Invocation

We instantiate Definitions 3 and 4 with maximum grade $G = 3$. All constructions are deterministic and use n parties with corruption threshold t.

B.1 Graded Consensus from Proxcensus

Binary s slot proxcensus places each honest output in a slot $q_i \in \{0, \ldots, s-1\}$, with all honest outputs in two adjacent slots and unanimous inputs 0 and 1 mapped to the two extreme slots. For odd $s = 2H + 1$, these slots correspond to $(0, H), (0, H-1), \ldots, (0, 1), (\perp, 0), (1, 1), \ldots, (1, H)$. Equivalently, the output consists of a bit and a grade in $\{0, \ldots, H\}$, with the bit immaterial at grade zero [FLZL21, Definition 2]. We map slot q to

$$\phi_H(q) = \begin{cases} (0, \min\{3, H - q\}), & q < H, \\ (\perp, 0), & q = H, \\ (1, \min\{3, q - H\}), & q > H. \end{cases}$$

For $t < n/3$, we use the three round, nine slot construction of Fitzi, Liu-Zhang, and Loss [FLZL21, Corollary 1] and apply ϕ_4. This construction uses only authenticated channels and requires no cryptographic setup. For $t < n/2$, we use their four round, seven slot construction [FLZL21, Lemma 3] and apply ϕ_3. The latter assumes unique threshold signatures requiring $n - t$ shares, with partial and combined signatures of size $O(\kappa)$. The input signatures and the round two endorsement signatures in that construction are bound to distinct message roles, for example $(\mathsf{sid}, \texttt{input}, v)$ and $(\mathsf{sid}, \texttt{endorse}, v)$. Here sid identifies the GC instance.

Lemma 18. *These constructions satisfy Definition 3 with $G = 3$ against the strongly rushing, adaptive adversary of our model. For $t < n/3$, GC takes 3 communication rounds and sends $O(n^2)$ bits; for $t < n/2$, it takes 4 communication rounds and sends $O(n^2\kappa)$ bits under the stated threshold signature assumption.*

Proof. Apply [FLZL21, Corollary 1 and Lemma 3], in the strongly rushing, adaptive model of its Section 2.1. The mapping ϕ_H preserves adjacency of grades and agreement of positive grade values, and maps unanimous inputs to grade 3.

B.2 A Compiler for Starts One Round Apart

We give an explicit round simulation for bounded start time differences, following the approach of synchronous protocol compilers [LS22,Nie03]. Let Π be a deterministic protocol consisting of $R \geq 1$ rounds of sending, receiving, and computing, with outputs produced after its last round. Messages are tagged by the instance identifier and the simulated round $j \in \{1, \ldots, R\}$. Party P_i, invoked at the beginning of physical round r_i, fixes its input and simulates round j as follows:

1. At the beginning of round $s_i(j) = r_i + 1 + 3(j-1)$, send the messages prescribed by simulated round j.
2. Collect round j messages received during physical rounds $s_i(j)-1$, $s_i(j)$, and $s_i(j)+1$. For each sender, retain the first correctly tagged message in that window, using the simulated protocol's convention for a missing message.
3. At the end of round $s_i(j)+1$, compute the simulated state transition, and output if $j = R$.

Messages are buffered from invocation onward, and those outside their specified windows are ignored. The initial idle round ensures that every honest party has been invoked before the first honest messages are sent.

Lemma 6. *Suppose Π is an R-round synchronous protocol and all honest parties invoke its staggered simulation in rounds r or $r+1$. The simulation preserves the correctness properties of Π, including against adaptive corruptions. If $r_{\max}$ is the latest honest invocation round, every honest party outputs by the end of round $r_{\max} + 3R - 1$, and honest output rounds differ by at most one. If Π sends at most B bits in at most M messages, the simulation sends at most $B + O(M \log(R+1))$ bits in at most M messages.*

Suppose all honest parties invoke the compiled protocol in rounds r or $r+1$ and at most t parties are corrupted. The simulation preserves the correctness of Π both without cryptographic setup and with idealized signatures, including against strongly rushing, adaptive corruptions. If $r_{\max}$ is the latest honest invocation round, every honest party outputs by the end of round $r_{\max} + 3R - 1$, and honest output rounds differ by at most one. If Π sends at most B bits in at most M messages, the compiled protocol sends at most $B + O(M \log(R+1))$ bits in at most M messages, excluding instance identifiers already required by Π.

Proof. For any two honest parties, $|s_i(j) - s_k(j)| \leq 1$. Hence every round j message sent by an honest party is delivered within every honest recipient's collection window. For $j = 1$, the earliest such delivery is no earlier than the latest invocation round, so no honest recipient misses an initial message.

Let $s_{\min}(j)$ and $s_{\max}(j)$ be the earliest and latest round j send times among honest parties. Every round j collection window closes by the end of round $s_{\max}(j) + 1 \leq s_{\min}(j) + 2$. The next honest send is at the beginning of round $s_{\min}(j+1) = s_{\min}(j) + 3$. Thus the adversary must fix every accepted round j message before seeing any honest message from simulated round $j+1$. Induction on j now identifies the accepted messages and state transitions with a synchronous execution of Π: honest messages are delivered correctly, and corrupted senders may choose arbitrary recipient specific messages or omissions within the same simulated round.

Adaptive corruptions and the suppression of their senders' messages are allowed in that synchronous execution. Equivalently for the deterministic correctness properties considered here, parties corrupted at any point may be treated as Byzantine throughout, reproducing their actual messages to the parties that

remain honest. In the authenticated case, the separation of collection windows from later sends also preserves the availability of honest signatures: a signature first generated in a later simulated round cannot enter an earlier one. This establishes the claimed preservation under the signature assumptions of Π.

Party P_i outputs at the end of round $r_i + 3R - 1 \leq r_{\max} + 3R - 1$. Since $r_i \in \{r, r+1\}$, honest output rounds differ by at most one. Each simulated message is sent once, with an additional $O(\log(R+1))$ bit round tag. This count also holds in executions with more than t corruptions whenever the bounds for Π do.

Corollary 1. *Applying Lemma 6 to Lemma 18 gives SGC with $G = 3$ and staggered termination bound 9 for $t < n/3$, using $O(n^2)$ bits, and bound 12 for $t < n/2$, using $O(n^2\kappa)$ bits.*

Proof. Use $R = 3$ and $R = 4$, respectively. The simulation preserves all graded consensus properties, and its constant size round tags preserve the asymptotic communication bounds.

C Cubic Honest Majority Consensus in $t + 2$ Rounds

The following protocol can be considered folklore as it follows easily from Dolev and Strong [DS83b]. Since there is, to the best of our knowledge, no publicly available description, we include a description and proof of correctness for completeness.

We consider a synchronous authenticated network of n parties $P_1, \ldots, P_n$, of which at most $t < n/2$ may be Byzantine. Each party P_i holds an input $v_i \in \{0,1\}^L$, and κ denotes the length of a signature.

All signatures are bound to the protocol instance through a session identifier sid. We distinguish *input signatures* from *forwarding signatures*: $\mathsf{inSig}_i(v) := \mathsf{Sign}_i(\mathsf{sid}, \mathsf{input}, v)$ and $\mathsf{fwdSig}_i(v) := \mathsf{Sign}_i(\mathsf{sid}, \mathsf{forward}, v)$. A forwarding signature is a signature on v itself, rather than on the message through which v was received; hence forwarding signatures received in different messages can be combined. An *input certificate* for a value v is a set $C_v = \{\mathsf{inSig}_j(v) : j \in I\}$ containing valid input signatures from $t + 1$ distinct parties.

The protocol, described in Figure 8, proceeds as follows. In round 0, every party disseminates its signed input. At the beginning of round 1, a party accepts up to two values for which it has an input certificate and broadcasts each certificate with its own forwarding signature. At the end of each round $r = 1, \ldots, t+1$, a party accepts a new value if it has an input certificate and at least r distinct forwarding signatures, provided it has accepted fewer than two values. For $r \leq t$, it appends its own forwarding signature and forwards the resulting set of $r + 1$ signatures in round $r+1$. A set of $t+1$ forwarding signatures contains an honest party's signature, ensuring that the value reaches every honest party by the end of round $t + 1$; a recipient that already holds two values need not accept it.

After processing messages received in round $t+1$, each party outputs v if its accepted set is the singleton $\{v\}$, and $\perp$ otherwise. For the binary BA instances used in committee escalation, we set $L = 1$ and map $\perp$ to 0.

Honest Majority Dolev-Strong Consensus

Each party P_i holds an input $v_i \in \{0,1\}^L$ and maintains a set A_i of accepted values, initially empty. Eligible values are processed in lexicographic order. Once $|A_i| = 2$, no further values are accepted, but all scheduled messages are still sent.

Round 0 **(input dissemination):** Each party P_i sends $\big(v_i, \mathsf{inSig}_i(v_i)\big)$ to every party.

Beginning of round 1**:** For each value $v \notin A_i$ for which P_i holds an input certificate C_v, and while $|A_i| < 2$, party P_i:

1. adds v to A_i;
2. computes $\mathsf{fwdSig}_i(v)$; and
3. sends $\big(v, C_v, \{\mathsf{fwdSig}_i(v)\}\big)$ to every party.

End of rounds $r = 1, \ldots, t+1$**:** Party P_i combines all valid forwarding signatures on each value v received through round r. For each value v, if party P_i has

1. a valid input certificate C_v,
2. forwarding signatures on v from at least r distinct parties,
3. $v \notin A_i$, and
4. $|A_i| < 2$,

then P_i adds v to A_i. If $r \leq t$, party P_i additionally computes $\mathsf{fwdSig}_i(v)$ and schedules $\big(v, C_v, \Sigma_v\big)$ for sending to every party in round $r+1$, where Σ_v contains $r+1$ distinct forwarding signatures on v, including $\mathsf{fwdSig}_i(v)$.

Decision (after the final acceptance step): Party P_i outputs

$$\mathsf{out}_i = \begin{cases} v, & \text{if } A_i = \{v\}, \\ \bot, & \text{otherwise.} \end{cases}$$

Fig. 8. An authenticated honest majority consensus protocol taking $t+2$ rounds; the binary BA instantiation maps $\bot$ to 0.

Correctness. For inputs in $\{0,1\}^L$, the protocol in Fig. 8 outputs in $\{0,1\}^L \cup \{\bot\}$. We prove strong validity, external validity for non-$\bot$ outputs, and agreement below; every honest party outputs and terminates after the final acceptance step in round $t+1$. The binary BA instantiation follows by setting $L = 1$ and mapping $\bot$ to 0.

Lemma 19 (Strong validity). *If all honest parties input the same value v, then every honest party outputs v.*

Proof. Since $n - t > t$, there are at least $t+1$ honest parties. Hence every honest party receives at least $t+1$ valid input signatures on v and obtains an input certificate for v.

For any $v' \neq v$, only Byzantine parties can produce input signatures on v'. Since there are at most t Byzantine parties, no input certificate for v' can exist. Therefore, v is the only value that can be accepted, and every honest party outputs v.

Lemma 20 (External validity). *Every non-$\perp$ output was input by an honest party.*

Proof. A value can be accepted only if it is accompanied by an input certificate containing signatures from $t+1$ distinct parties. At least one of these parties is honest, and an honest party produces an input signature only on its own input value.

Lemma 21 (Propagation). *If an honest party accepts a value v, then, by the end of round $t+1$, every honest party either accepts v or accepts two distinct values.*

Proof. If an honest party accepts v at the beginning of round 1, it sends its input certificate and forwarding signature to every party in that round. Thus every honest party accepts v or holds two accepted values by the end of round 1.

Suppose next that an honest party P_i accepts v at the end of round r, where $1 \leq r \leq t$. It has an input certificate and at least r distinct forwarding signatures on v. Since P_i has not previously accepted v, it has not previously produced a forwarding signature on v. It therefore appends its own signature to obtain $r+1$ distinct forwarding signatures and sends them with the input certificate to every party in round $r+1$. This send occurs even if P_i has accepted a second value in the meantime. Consequently, every honest party accepts v or holds two accepted values by the end of round $r+1 \leq t+1$.

Finally, suppose that an honest party accepts v at the end of round $t+1$. It has at least $t+1$ distinct forwarding signatures on v, including a signature of an honest party P_j. Party P_j produced that signature either at the beginning of round 1 or after accepting v at the end of a round $r \leq t$. In both cases, the preceding arguments show that every honest party accepts v or holds two accepted values by the end of round $t+1$.

Theorem 3 (Agreement). *All honest parties output the same value.*

Proof. Suppose an honest party P_i outputs a value $v \neq \perp$. Then $A_i = \{v\}$.

By the propagation lemma, every honest party either accepts v or accepts at least two distinct values. Suppose, toward a contradiction, that some honest party accepts two distinct values and therefore outputs $\perp$. Let $v' \neq v$ be one of its accepted values. Applying the propagation lemma to v', party P_i must either accept v' or accept at least two values. Both contradict $A_i = \{v\}$.

Hence every honest party accepts v and no other value, and therefore every honest party outputs v. If no honest party outputs a non-$\perp$ value, then all honest parties output $\perp$.

Complexity. The protocol consists of one input dissemination round followed by $t+1$ forwarding rounds, for a total of $t+2$ rounds.

Each honest party disseminates its input and forwards at most two distinct values, sending each message to all n parties. Hence honest parties send $O(n^2)$ messages in total. A forwarding message contains:

- an L-bit value;
- $t+1$ input signatures; and
- at most $t+1$ forwarding signatures.

Its size is therefore $O(L+(t+1)\kappa)$, and the total communication of honest parties is $O\big(n^2(L+(t+1)\kappa)\big)$ bits. For binary inputs, this is $O(n^3\kappa)$ bits, including the case $t=0$.

Corollary 2 (Binary Byzantine agreement). *For $t < n/2$, setting $L = 1$ in Fig. 8 and mapping $\perp$ to 0 satisfies Definition 2, with $t+2$ termination rounds and $O(n^3\kappa)$ communicated bits.*

Proof. Agreement is preserved by applying the same output map at every party. If all honest parties input the same bit v, strong validity gives output v before the map is applied, so binary validity holds. The output map adds no communication or rounds, and every honest party terminates after rounds $0, \ldots, t+1$.

D Detailed Security Analysis

D.1 Security of an Uninterrupted Node

The lemmas in this subsection hold for every schedule s fixed above and concern a good invocation entered by every honest member of its committee. Claims about outputs apply before an honest party terminates an ancestor invocation; completion claims assume that no such termination occurs first. At termination, the party cancels all unfinished subprotocols it invoked, recursively, as specified in the preliminaries. The following property ensures that every honest party has obtained the current invocation's output before such cancellation begins.

Lemma 22 (Output and termination of a recursive invocation). *Consider a good invocation of RCE_s entered by every honest member of its committee, and assume that no honest party cancels it because of ancestor termination before it terminates. At an internal node, an honest party that outputs in round r terminates at the end of round $r+1$. At a leaf, honest parties output and terminate in the final round of the fixed round BA instance. In either case, every honest party has output by the first honest termination of the invocation.*

Proof. At an internal node, output and termination occur at the times specified by the output rule in Fig. 5. If the first honest output occurs in round r, no honest party terminates the invocation before the end of round $r+1$. Output staggering of $\mathsf{Pulse}^{\mathsf{out}}$ therefore gives every honest party its output by that time. At a leaf, all honest parties start the fixed round BA together and output and terminate in its final round. The leaf claim uses these concrete BA instantiations, rather than an output timing guarantee of abstract BA.

Thus, to prove claims about outputs, we consider the execution before the first honest termination of the current or an ancestor invocation, while every

honest party still participates. A subprotocol cancelled at that point need not output; the terminating caller has already produced the result needed by its own caller.

Once honest output values are known to agree, it suffices for completion to show that, if no honest party outputs before round r, every honest party supplies that value to $\mathsf{Pulse}^{\mathtt{out}}$ by the beginning of round r. An earlier honest output makes every honest party output by the end of round r, by output staggering; otherwise, pulse termination gives the same bound. Every honest party then terminates by the end of round $r + 1$.

We start by showing that if all honest parties have the same input, then this is the only possible output.

Lemma 23 (Maintaining honest preagreement). *If all honest inputs to* $\mathsf{RCE}_s(C, k)$ *are* v*, then, for every honest party with output* $y_i \neq \perp$*:* $y_i = v$*.*

Proof. If $|C| \leq c(s(k) + 1)$, the node is a leaf and the protocol invokes $\mathrm{BA}(C)$. Since every honest party participating in this BA instance has input v, validity of Byzantine agreement (Definition 2) makes every honest output equal to v.

Suppose now that $|C| > c(s(k)+1)$, so the node is an internal node. By Line 1, every honest party P_i invokes GC^0 with input $x_i^0 = x_i = v$. By graded validity of graded consensus (Definition 3), every honest output from GC^0 is $(v, 3)$ and therefore supplies input v to $\mathsf{Pulse}^{\mathtt{out}}$. Each honest party then sets $b_i^{\mathtt{out}} = \top$, so the later guards prevent another input to $\mathsf{Pulse}^{\mathtt{out}}$. Pulse validity (Definition 5) therefore implies that every honest party with output $y_i \neq \perp$ has $y_i = v$.

We next establish the value propagation facts used for agreement when honest parties obtain their input to $\mathsf{Pulse}^{\mathtt{out}}$ from different steps of the protocol. Recall that the king committee C^0 and the two child committees C^1, C^2 partition the committee of the current invocation.

Lemma 24 (Completion after grade 3). *If an honest party outputs* $(v, 3)$ *from* GC^0 *or* SGC^1 *in round* r*, then every honest output from the current invocation equals* v*. Every honest party outputs by the end of round* $r + \delta_{\mathsf{SGC}} + 3$ *and terminates by the end of round* $r + \delta_{\mathsf{SGC}} + 4$*.*

Proof. Suppose first that an honest party outputs $(v, 3)$ from GC^0. By graded agreement (Definition 3), every honest output from GC^0 has value v and grade at least 2. Hence every honest input to Pulse^0 equals v, and its validity and termination (Definition 5) ensure that every honest input to SGC^1 equals v. Graded validity (Definition 4) makes every honest output from SGC^1 equal to $(v, 3)$. Every honest input to $\mathsf{Pulse}^{\mathtt{out}}$ in Line 2 or Line 7 therefore equals v. Line 7 sets $b_i^{\mathtt{out}} = \top$ for every honest party that has not already set it, so the later guards prevent another honest input.

Now suppose that an honest party outputs $(v, 3)$ from SGC^1. If an honest party previously output $(v', 3)$ from GC^0, the first case makes every honest input to SGC^1 equal to v', so graded validity gives $v = v'$ and the first case applies. We may therefore assume that no honest party inputs a value to $\mathsf{Pulse}^{\mathtt{out}}$ in Line 2.

Graded agreement (Definition 4) implies that every honest output from SGC^1 has value v and grade at least 2. The same argument using Pulse^1 and SGC^2 makes every honest input to $\mathsf{Pulse}^{\mathtt{out}}$ equal to v, and Line 13 sets the guards before the right child threshold can supply another input.

In both cases, pulse validity implies $y_i = v$ for every honest party with $y_i \neq \perp$.

For completion, consider the steps taken while no honest party has output from the current invocation. The chosen GC and SGC instances have honest output rounds differing by at most one, so all inputs to the next phase pulse are supplied by the beginning of round $r + 2$. That pulse outputs by the end of round $r + 2$, and the next SGC starts by round $r + 3$ and outputs by the end of round $r + \delta_{\mathsf{SGC}} + 2$. All honest inputs to $\mathsf{Pulse}^{\mathtt{out}}$ are therefore supplied by the beginning of round $r + \delta_{\mathsf{SGC}} + 3$, and the completion argument above gives the stated bounds.

Lemma 7 (Good King Committee). *If C^0 is good, then all honest outputs from the current invocation agree. Every honest party outputs by the end of round $r_1 + \delta_{\mathsf{SGC}}$ and terminates by the end of the following round.*

Proof. If some honest party outputs $(v, 3)$ from GC^0, then Lemma 24 proves the claim. Suppose therefore that every honest party has grade less than 3.

If some honest party outputs (v, g) with $g \geq 2$ from GC^0, graded agreement makes every honest king member input v to BA, so every honest BA output equals v by validity. Since C^0 is good, every threshold in Line 3 includes an honest king output, and these outputs agree by BA agreement. Thus all honest inputs to Pulse^0 agree, including when both direct inputs from graded consensus and threshold inputs occur.

Write w for their common value. Pulse validity makes every input to SGC^1 obtained through Pulse^0 equal to w. The timeout cannot introduce another value: if round r_1 is reached, the good king BA has completed and supplied all remaining inputs to Pulse^0 in time for its outputs to precede that round. Thus every honest input to SGC^1 equals w, and graded validity and Lemma 24 give agreement on w.

For completion, consider the steps taken while no honest party has output from the current invocation. The king BA finishes by the end of round $r_0 + \delta_{\mathsf{GC}} + \delta_{\mathsf{BA}(C^0)} - 1$. Its honest members number at least $|C^0|/c$ and disseminate their common output in the next round, so all remaining inputs to Pulse^0 are supplied by the beginning of round $r_1 - 1$. All honest parties therefore invoke SGC^1 by round r_1, obtain $(w, 3)$ by the end of round $r_1 + \delta_{\mathsf{SGC}} - 1$, and supply w to $\mathsf{Pulse}^{\mathtt{out}}$ by the beginning of round $r_1 + \delta_{\mathsf{SGC}}$. The completion argument above gives the stated bounds. □

Lemma 8 (Compatibility with a child output). *Let $j \in \{1, 2\}$ and suppose that C^j is good. If an honest member of C^j outputs w from the recursive invocation executed by C^j and an honest party outputs (v, g) from SGC^j with $g \geq 2$, then $w = v$.*

Proof. The graded agreement and termination guarantees in Definition 4 imply that every honest party outputs (v, g_i) from SGC^j for some $g_i \geq 1$. Hence every

honest member of C^j invokes the recursive escalation with input v in Line 8 when $j = 1$ and Line 14 when $j = 2$. Since this invocation is good, Lemma 23 implies that every honest member that outputs from it outputs v. In particular, the honest member that outputs w must have $w = v$.

Lemma 9 (Good Child Committee). *Let $j \in \{1, 2\}$. Suppose that C^j is good and every honest member of C^j outputs w from its recursive invocation by the end of round $r_j + \delta_{\mathsf{SGC}} + D_s(|C^j|, k_j) - 1$, where $k_1 = k + 1$ and $k_2 = k_R$. Suppose that no honest party inputs to $\mathsf{Pulse}^{\mathtt{out}}$ in Line 2 or, when $j = 2$, in Line 7. Then every honest output from the current invocation equals w, and every honest party outputs and terminates.*

Proof. Since C^j is good, every value received from at least $|C^j|/c$ members includes an honest child output and therefore equals w. By Lemma 8, every honest output of grade at least 2 from SGC^j also has value w.

Consider first $j = 1$. Every honest input to Pulse^1, and every input supplied to $\mathsf{Pulse}^{\mathtt{out}}$ in Line 7, equals w. Pulse validity makes every input to SGC^2 obtained through Pulse^1 equal to w. The child output deadline is $r_2 - 3$, leaving two rounds for dissemination and the pulse before the timeout in round r_2. Thus the timeout cannot introduce another value, and every honest input to SGC^2 equals w. Graded validity makes every honest output from SGC^2 equal to $(w, 3)$, so every honest input to $\mathsf{Pulse}^{\mathtt{out}}$ in Line 13 equals w. This line sets $b_i^{\mathtt{out}} = \top$ for every honest party that has not already set it, and the guard in Line 15 prevents another input.

Now consider $j = 2$. The preceding compatibility argument shows that both the grade 3 inputs in Line 13 and the threshold inputs in Line 15 equal w. In both cases, every honest input to $\mathsf{Pulse}^{\mathtt{out}}$ equals w, so pulse validity gives $y_i = w$ for every honest party with $y_i \neq \perp$.

For completion, consider the steps taken while no honest party has output from the current invocation. Every honest child member disseminates w after obtaining it, and at least $|C^j|/c$ members are honest. For $j = 1$, all remaining inputs to Pulse^1 are supplied by the beginning of round $r_2 - 1$, so all honest parties invoke SGC^2 by round r_2 and supply w to $\mathsf{Pulse}^{\mathtt{out}}$ by the beginning of round $r_2 + \delta_{\mathsf{SGC}}$. For $j = 2$, all remaining inputs to $\mathsf{Pulse}^{\mathtt{out}}$ are supplied by the beginning of round $r_2 + \delta_{\mathsf{SGC}} + D_s(|C^2|, k_R) + 1$, two rounds after the last allotted child output round. The completion argument above gives all honest outputs by the respective round and termination by the end of the following round.

D.2 Round Complexity

For this subsection and the first two instantiations, set $s(k) = k$ and write RCE, N, and D for RCE_s, N_s, and D_s, respectively.

We now analyze the termination time as a function of the actual number f of corrupted parties. Throughout the analysis, fix the corruption set F, with $|F| = f < n/c$. The preceding subsection considered good invocations, whereas the full recursion tree is guaranteed to have a good committee only at its root.

For a node u, let C_u denote its committee, let C_u^0, C_u^1, C_u^2 denote its king, left child, and right child subcommittees, let $n_u := |C_u|$, and let k_u denote its preorder index.

The king committee of an internal node u has size $c \cdot k_u$, and its local Byzantine agreement instance has duration $k_u + O(1)$. All graded consensus, pulse, and local processing steps in each internal node contribute only a constant number of additional rounds per node. Leaves require separate accounting, since a leaf u with index k_u may have only $n_u \ll c \cdot k_u$ parties assigned to it. Each leaf runs an instance of BA which tolerates $t_u = \lceil n_u/c \rceil - 1$ corruptions and it will consequently run for $t_u + O(1)$ rounds.

Bounding the size and deadline of a subtree. We first bound the number of nodes and deadline of an arbitrary complete subtree. Note that both of these are used in the protocol description as functions N and D of the preorder index and number of parties in the node. For convenience we will use N_u and D_u in the following to refer to the outputs of these functions evaluated on the parameters of node u. Thus, $N_u := N(n_u, k_u)$ and $D_u := D(n_u, k_u)$. We first bound N_u.

Lemma 25 (Number of nodes in a subtree). *The subtree rooted at a node u with n_u parties contains $O(\sqrt{n_u/c + 1})$ nodes.*

Proof. Let I_u and L_u be the numbers of internal nodes and leaves in the subtree rooted at u. The king committees of its internal nodes are disjoint subsets of C_u, and each contains ck_v parties. Their indices are distinct positive integers, so their total size is at least $c(1 + 2 + \cdots + I_u) = cI_u(I_u + 1)/2$. It follows that $I_u(I_u + 1) \leq 2n_u/c$ and hence $I_u = O(\sqrt{n_u/c + 1})$. Every internal node has two children, so counting edges gives $2I_u = I_u + L_u - 1$. Thus $L_u = I_u + 1$ and $N_u = 2I_u + 1 = O(\sqrt{n_u/c + 1})$.

Next, we bound D_u: the fixed deadline for the subtree rooted at u, obtained by pessimistically allowing every recursive child to use its worst case round complexity.

Lemma 26 (Deadline of a subtree). *For every node u, the deadline satisfies $D_u \leq n_u/c + O(N_u) \leq n_u/c + O(\sqrt{n_u/c + 1})$.*

Proof. A leaf v contributes at most $\lceil n_v/c \rceil + O(1) = n_v/c + O(1)$ rounds to the deadline. An internal node v contributes its king BA and the local graded consensus, pulse, and message passing steps, for at most $|C_v^0|/c + O(1) = k_v + O(1)$ rounds. Expanding the deadline recurrence sums these contributions over the subtree. The king committees of the internal nodes and the leaf committees partition C_u, so their size dependent contributions sum to n_u/c. The remaining constant cost at each node gives $D_u \leq n_u/c + O(N_u)$, and Lemma 25 bounds N_u by $O(\sqrt{n_u/c + 1})$.

Worst case path and completion. We next define a worst case path through the recursion tree, which we use to upper bound the running time with respect

to a fixed set of corrupted parties F. The intuition behind this worst case path is that it describes the leftmost path in the tree which consists of only good nodes and ends as soon as it finds a good king committee.

Definition 6 (Worst case path). *Fix a set of corrupted parties F. Define the worst case path $u_1, \ldots, u_r$ by letting u_1 be the root. If a node u_i lies on the path, the path continues as follows:*

- ***End:*** *If u_i is a leaf or $C^0_{u_i}$ is good, the path ends at u_i.*
- ***Left transition:*** *If $C^0_{u_i}$ is bad, and $C^1_{u_i}$ is good, the next node on the path, u_{i+1} is the left child of u_i.*
- ***Right transition:*** *If both $C^0_{u_i}$ and $C^1_{u_i}$ are bad, the next node on the path, u_{i+1} is the right child of u_i.*

We say that u_r is the terminal node and that each preceding node is nonterminal.

The path follows increasing preorder indices and ends at a leaf or an internal node with a good king committee. For an output obtained at the terminal node to propagate consistently through the recursive callers, every node on the path must be good. We first show that all nodes on the worst case path are good (have less than a $1/c$ fraction corrupted parties).

Lemma 27. *Let F be the set of faulty parties and $u_1, \ldots, u_r$ the corresponding worst case path. For each node u_i on the path, $|C_{u_i} \cap F| < |C_{u_i}|/c$.*

Proof. This is by definition true for the root, since $|F| = f < n/c$, and for any left child added, since the path only enters a left child when it is good. It is also true for any right child added: if the king committee and left child of a good node u are bad, then the right child must be good. Since C^0_u, C^1_u, C^2_u partition C_u, if all three were bad, then the number f_u of corruptions in C_u would satisfy $f_u \geq \frac{|C^0_u|}{c} + \frac{|C^1_u|}{c} + \frac{|C^2_u|}{c} = \frac{|C_u|}{c}$, contradicting that C_u is good.

For each suffix $u_i, \ldots, u_r$ of the worst case path, define its path schedule S_i backward from the terminal node. If u_r is a leaf, let $S_r = T_{\mathsf{BA}}(n_{u_r})$; otherwise, let $S_r = T_{\mathsf{BA}}(c \cdot k_{u_r}) + d_{\mathrm{loc}}$. For every $i < r$, let

$$S_i = \begin{cases} T_{\mathsf{BA}}(c \cdot k_{u_i}) + d_{\mathrm{loc}} + S_{i+1} & \text{for a left transition,} \\ T_{\mathsf{BA}}(c \cdot k_{u_i}) + d_{\mathrm{loc}} + D_{v_i} + S_{i+1} & \text{for a right transition,} \end{cases}$$

where v_i is the left child skipped by a right transition.

Lemma 28 (Worst case path bounds completion). *There is a value w such that every honest party P_i sets $y_i = w$ and terminates the top level invocation within S_1 rounds.*

Proof. Every node on the path is good by Lemma 27. We prove backward along the path that each invocation has agreement and completes within S_i rounds of its starting round, with the qualifications in Section D.1. We simultaneously

observe that $S_i \le D_{u_i}$. The terminal node is either a leaf or has a good king committee. At a leaf, the claims follow from agreement and termination of BA (Definition 2), and $S_r = D_{u_r}$. If the terminal node has a good king committee, Lemma 7 proves the claims within $S_r = T_{\mathsf{BA}}(c \cdot k_{u_r}) + d_{\mathrm{loc}}$ rounds. The definition of D_{u_r} includes this local work, so $S_r \le D_{u_r}$.

Now consider a nonterminal node u_i and assume the claims for u_{i+1}. If an honest party inputs to $\mathsf{Pulse}^{\mathtt{out}}$ in Line 2 or Line 7, then it obtained grade 3 from GC^0 or SGC^1, respectively. Lemma 24 proves agreement and termination, and the local steps used in its proof are covered by d_{loc}. Suppose henceforth that no honest party supplies either of these inputs. Let $j = 1$ for a left transition and $j = 2$ for a right transition, so $C^j_{u_i} = C_{u_{i+1}}$ is good. Consider the honest inputs to Pulse^1 when $j = 1$, or to $\mathsf{Pulse}^{\mathtt{out}}$ when $j = 2$. The direct inputs from SGC^j agree by graded agreement. Each threshold input includes a value sent by an honest member of $C^j_{u_i}$ after it output from the recursive invocation, and those outputs agree by the induction hypothesis. When both sources occur, Lemma 8 equates their values. Thus all honest inputs to the respective pulse agree, including before every honest child member has output.

First suppose that the path makes a left transition, so $C^1_{u_i} = C_{u_{i+1}}$ is good. The definition of the path schedules and the induction hypothesis give

$$S_i = T_{\mathsf{BA}}(c \cdot k_{u_i}) + d_{\mathrm{loc}} + S_{i+1} \le T_{\mathsf{BA}}(c \cdot k_{u_i}) + d_{\mathrm{loc}} + D_{u_{i+1}} \le D_{u_i}.$$

Every honest input to SGC^2 supplied before the fixed starting round of the Right phase comes from Pulse^1, so these inputs agree by pulse validity. If that starting round is reached, the induction hypothesis guarantees every honest child output by its deadline. The dissemination round and pulse termination therefore make all honest outputs from Pulse^1 available before the timeout, as in Lemma 9. Hence all honest inputs to SGC^2 agree, and graded validity gives grade 3 on that common value. The guards prevent later honest inputs from the right child threshold, so all honest inputs to $\mathsf{Pulse}^{\mathtt{out}}$ agree.

Now suppose that the path makes a right transition. Let v_i be the skipped left child of u_i. Since $C^0_{u_i}$ and $C^1_{u_i}$ are bad and u_i is good, $C^2_{u_i} = C_{u_{i+1}}$ is good. By the definition of r_2, the Right phase starts at its fixed round after the full deadline D_{v_i} allotted to the left subtree unless Pulse^1 starts it earlier. The definition of the path schedules and the induction hypothesis give

$$S_i = T_{\mathsf{BA}}(c \cdot k_{u_i}) + d_{\mathrm{loc}} + D_{v_i} + S_{i+1} \le T_{\mathsf{BA}}(c \cdot k_{u_i}) + d_{\mathrm{loc}} + D_{v_i} + D_{u_{i+1}} = D_{u_i}.$$

Here the preceding compatibility argument directly gives agreement of all honest inputs to $\mathsf{Pulse}^{\mathtt{out}}$.

In both cases, pulse validity proves agreement. For completion, let r_0 be the starting round of this invocation and set $R = r_0 + S_i - 2$. If no honest output occurs before round R, all honest parties continue to participate, so the selected good child is invoked at its fixed starting round and every honest member outputs within S_{i+1} rounds by induction. The local steps in Lemma 9, whose delays are included in d_{loc}, then supply every honest input to $\mathsf{Pulse}^{\mathtt{out}}$ with the common value by the beginning of round R. The completion argument in Section D.1

therefore gives every honest output by the end of round R and termination by the end of round $R+1 = r_0 + S_i - 1$.

At the root there is no ancestor invocation, so these guarantees prove the lemma.

Bounding the preorder index of the terminal node. We now bound the terminal index by charging every transition on the worst case path to corrupted parties. Each left transition can be paid for by the bad king committee of its parent.

Lemma 11 (Corruption cost of a left transition). *Suppose the path transitions from u_i to its left child. Then the king committee $C^0_{u_i}$ is bad and contains at least k_{u_i} corrupted parties.*

Proof. The Left case of the path definition requires $C^0_{u_i}$ to be bad. Since $|C^0_{u_i}| = c \cdot k_{u_i}$, it contains at least $|C^0_{u_i}|/c = k_{u_i}$ corrupted parties.

We then show that a right transition can be paid for by corruptions in the parent king committee and the skipped left subtree. In particular, when the left child is bad, we pessimistically assume that it produces no useful output and wait for its full deadline before proceeding to the right child. Any output that a bad component produces in an actual execution can only accelerate termination.

Lemma 12 (Corruption cost of a right transition). *Suppose the path transitions from u_i to its right child u_{i+1}. Then $C^0_{u_i} \cup C^1_{u_i}$ contains $\Omega(k^2_{u_{i+1}} - k^2_{u_i})$ corrupted parties.*

Proof. Write $k_i := k_{u_i}$ and $k_{i+1} := k_{u_{i+1}}$, and let I_i be the number of internal nodes in the skipped left subtree rooted at v_i. Both $C^0_{u_i}$ and C_{v_i} are bad, so they contain at least k_i and n_{v_i}/c corruptions, respectively. The internal nodes of the skipped subtree have distinct indices at least k_i+1, and their king committees are disjoint subsets of C_{v_i}. Hence $n_{v_i}/c \geq I_i k_i + I_i(I_i+1)/2$. Together, the bad king committee and skipped subtree therefore contain at least $(I_i+1)k_i + I_i(I_i+1)/2$ corruptions.

The skipped subtree has $2I_i+1$ nodes, so $k_{i+1} - k_i = 2(I_i+1)$. Consequently, $k^2_{i+1} - k^2_i = 4(I_i+1)k_i + 4(I_i+1)^2$. Since $k_i \geq 1$, the corruption bound above is $\Omega((I_i+1)k_i + (I_i+1)^2)$, proving the claim.

Lemma 13 (Preorder index of the terminal node). *The terminal node u_r satisfies $k_{u_r} = O(\sqrt{f})$.*

Proof. Write $k_i := k_{u_i}$. For a left transition, $k_{i+1} = k_i+1$, so $k^2_{i+1} - k^2_i = 2k_i+1 = O(k_i)$. By Lemma 11, the bad king committee contains at least k_i corruptions, which pay for this increase. For a right transition, Lemma 12 charges the increase $k^2_{i+1} - k^2_i$ to corruptions in the king committee and skipped left subtree. These charged sets are pairwise disjoint, so summing over the path gives $k^2_r - k^2_1 = O(f)$. Since $k_1 = 1$, we obtain $k_{u_r} = O(\sqrt{f+1})$.

Theorem 2 (Early stopping round complexity). *If an execution contains $f < n/c$ corrupted parties, all honest parties terminate within $f+O(\sqrt{f})$ rounds.*

Proof. By Lemma 28, it suffices to bound S_1. At each nonterminal path node u_j, the king BA takes at most $k_{u_j} + O(1)$ rounds, and its bad committee contains at least k_{u_j} corruptions. Let $\mathcal{B}$ be the collection of bad left children skipped by right transitions, and write $f_v := |F \cap C_v|$ for $v \in \mathcal{B}$. Since $f_v \geq n_v/c$, Lemma 26 gives $D_v \leq f_v + O(N_v)$. The skipped subtrees and the king committees on the path are disjoint, so $\sum_{v\in\mathcal{B}} f_v + \sum_{j=1}^{r-1} k_{u_j} \leq f$. Thus the size dependent terms in these deadlines contribute at most f rounds.

Write $K := k_{u_r} = O(\sqrt{f+1})$, using Lemma 13. Skipping a subtree rooted at v increases the preorder index by $1 + N_v$, so $\sum_{v\in\mathcal{B}} N_v \leq K$. There are also at most K nodes on the path, each contributing a constant number of local rounds. The overhead from these nodes and the skipped subtrees is therefore $O(K)$.

If u_r is internal, its king BA and local steps take at most $K+O(1)$ rounds. If it is a leaf, then $n_{u_r} \leq c(K+1)$, so its BA takes at most $\lceil n_{u_r}/c \rceil + O(1) \leq K + O(1)$ rounds. Adding this terminal cost gives $S_1 \leq f + O(K) = f + O(\sqrt{f+1})$.

Corollary 3 (Validity). *The protocol of Fig. 5 satisfies validity as defined in Definition 2.*

Proof. Every honest party participates in the top level invocation. By Lemma 23, every honest party with output $y_i \neq \perp$ has the common honest input as its value, and Theorem 2 guarantees that every honest party terminates.

Corollary 4 (Agreement). *The protocol of Fig. 5 satisfies agreement as defined in Definition 2.*

Proof. The claim is the agreement conclusion of Lemma 28.

D.3 Communication Complexity

Let $\mathsf{B}(m)$ be a nondecreasing upper bound on the sum of the local sending budgets of the abstract Byzantine agreement protocol when it is run by m parties. The sending limits specified in the preliminaries ensure that this bound also holds in bad instances. We account separately for these BA instances and for the remaining graded consensus, pulse, and local message passing logic. An authenticated pulse among m members of a node committee communicates $O(m^2\kappa)$ bits by Lemma 15. Each node invokes a constant number of GC, SGC, and pulse instances and disseminates subcommittee outputs to its n_u parties, for $O(n_u^2)$ bits in the setting without cryptographic setup and $O(n_u^2\kappa)$ bits in the authenticated setting. The two child committees have size at most $n_u/2$, so the sum of their squared sizes is at most $n_u^2/2$. Summing over the recursion tree gives $O(n^2)$ and $O(n^2\kappa)$, respectively. The cost of identifying protocol instances is included in these bounds. The channel identifies the sender, and each party belongs to at most one committee at each depth, so local messages need only specify their depth and phase. Including $O(d+1)$ bits for this identifier at depth d gives

$O(n^2 \sum_{d\geq 0}(d+1)/2^d) = O(n^2)$ bits, or $O(n^2\kappa)$ in the authenticated setting. BA instances have disjoint scheduled intervals, so their identifiers can be recovered from the public schedule.

Consider an execution with f corrupted parties, whose terminal path node u_r has preorder index k_{u_r}. Recursive calls follow pre-order: the complete deadline of a left subtree precedes the fixed start of its right sibling. By the completion bounds in Lemmas 7 and 9, a recursive call begun while the terminal output propagates is stopped with its caller by the end of the following round. Only a leaf can begin BA in this time, since an internal call first runs GC for at least three rounds. The caller produces all its outputs at least three rounds before its own deadline, so the same bounds prevent further sibling calls from starting at higher ancestors. Thus every BA instance begun before termination lies in the preorder prefix ending at u_r, except possibly a leaf at index $k_{u_r}+1$ begun while the terminal output propagates. Such a leaf has at most $c(k_{u_r}+2)$ parties. Hence $k_{u_r} = O(\sqrt{f+1})$ bounds both the number of BA instances and their committee sizes up to constant factors.

At an internal node of index $k \leq k_{u_r}+1$, the king BA has ck parties; at a leaf, its committee has at most $c(k+1)$ parties. Thus every BA instance has at most $M_f = O(\sqrt{f+1})$ parties, and monotonicity of B bounds their total communication by $O(\sqrt{f+1}\,\mathsf{B}(M_f))$. Including the local communication gives $O(n^2 + \sqrt{f+1}\,\mathsf{B}(M_f))$ bits in the setting without cryptographic setup. If $n \leq 3t$, then $n^2 = O(n(t+1))$. If $n > 3t$, fix $C = (P_1, \ldots, P_{3t+1})$, run the protocol among C, and let every member of C send its output to all parties. Every remaining party outputs v upon receiving v from $t+1$ members of C. At least $2t+1$ honest members send the agreed output, whereas at most t members can support another value, so validity and agreement are preserved and termination takes one additional round. The execution within C contains at most f corruptions, and its $O(|C|^2)$ non-BA communication plus dissemination is $O((t+1)^2 + n(t+1)) = O(n(t+1))$.

In the authenticated setting, this gives $O(n(t+1)\kappa)$ non-BA bits, and the BA protocol of Section C has $\mathsf{B}(m) = O(m^3\kappa)$. The BA instances therefore contribute $O(\sqrt{f+1}\,M_f^3\kappa) = O((f+1)^2\kappa)$ bits, which are absorbed by $O(n(t+1)\kappa)$. The total authenticated communication is consequently $O(n(t+1)\kappa)$ bits.

Without cryptographic setup, we use the BA protocol of [KM13]. Its communication satisfies $\mathsf{B}(m) = O(m^3 \log m)$. Its BA contribution is $O((f+1)^2 \log(f+2))$ bits, giving $O(n(t+1) + f^2 \log(f+2))$ bits in total after absorbing the constant cases into $O(n(t+1))$.

D.4 Optimal Communication Supermajority Instantiation

Set $s(k) := \lceil k/\log_2(k+1) \rceil$.

Lemma 29. *The function s is nondecreasing, satisfies $s(k) \leq k$, and for every $K \geq 1$,*

$$\sum_{k=1}^{K} s(k) = \Theta\left(\frac{K^2}{\log(K+1)}\right), \qquad s(2K) = O(s(K)). \tag{1}$$

Proof. The function $k/\log_2(k+1)$ is increasing for $k \geq 1$. The definition gives $s(k) = \Theta(k/\log(k+1)+1)$. Integral comparison gives the sum, and substituting $2K$ gives $s(2K) = O(s(K))$.

Set $c = 3$ in Fig. 5, and write $\mathsf{RCE}^\star$, $N^\star$, and $D^\star$ for RCE_s, N_s, and D_s, respectively.

Lemma 30. *Write $N_u^\star := N^\star(n_u, k_u)$ and $D_u^\star := D^\star(n_u, k_u)$. For every subtree rooted at a node u, we have $N_u^\star = O(n_u)$ and $D_u^\star \leq n_u/3 + O(N_u^\star)$.*

Proof. If the subtree has I internal nodes, it has $I+1$ leaves and $N_u^\star = 2I+1$. Each internal node removes at least three parties, so $I = O(n_u)$ and hence $N_u^\star = O(n_u)$. The partition argument in Lemma 26 applies to every schedule s, giving $D_u^\star \leq n_u/3 + O(N_u^\star)$.

Define the worst case path for $\mathsf{RCE}^\star$ by Definition 6 using its recursion tree and committees. For its terminal node u_r, set $S_r^\star = T_{\mathsf{BA}}(n_{u_r})$ if it is a leaf, and $S_r^\star = T_{\mathsf{BA}}(3s(k_{u_r})) + d_{\mathrm{loc}}$ otherwise. For $i < r$, set $S_i^\star = T_{\mathsf{BA}}(3s(k_{u_i})) + d_{\mathrm{loc}} + S_{i+1}^\star$ for a left transition. For a right transition, add the skipped left subtree's deadline $D^\star(|C_{u_i}^1|, k_{u_i}+1)$ to this expression.

Lemma 31. *Every node on this path is good, and all honest parties terminate the top level invocation with a common output within $S_1^\star$ rounds.*

Proof. Every node on the path is good by the argument of Lemma 27, which uses only the committee partition and the rules for choosing the path. The single node guarantees in Lemmas 7 to 9, 23 and 24 also hold for every fixed schedule s. We can therefore apply the backward induction of Lemma 28, using $T_{\mathsf{BA}}(3s(k_{u_i}))$ for each king BA and $D^\star(|C_{u_i}^1|, k_{u_i}+1)$ for each skipped left subtree. These are the durations defining $S_i^\star$, so the root invocation terminates with a common output within $S_1^\star$ rounds.

Lemma 32. *The terminal index $K := k_{u_r}$ is $O(\sqrt{(f+1)\log(f+2)})$.*

Proof. For a left transition from index k, the sum $\sum_{h=1}^{k} s(h)$ increases by $s(k+1) = O(s(k))$. The bad king committee contains at least $s(k)$ corruptions, which pay for this increase.

For a right transition, let I be the number of internal nodes in the skipped left subtree. Its internal nodes have distinct indices at least $k+1$, so their disjoint king committees contain at least $3\sum_{h=1}^{I} s(k+h)$ parties. Since the skipped subtree is bad, it contains at least $\sum_{h=1}^{I} s(k+h)$ corruptions, in addition to the $s(k)$ corruptions in the parent's king committee. The next index is $k+2I+2$, because the skipped subtree has $2I+1$ nodes. For $0 \leq j \leq I$, monotonicity and $s(2x) = O(s(x))$ give $s(k+2j+1), s(k+2j+2) = O(s(k+j))$; when $k=1$, apply the doubling bound once more. Pairing these terms shows that the increase $\sum_{h=k+1}^{k+2I+2} s(h)$ is $O(s(k) + \sum_{h=1}^{I} s(k+h))$ and can therefore be charged to those corruptions.

All charged committees and skipped subtrees are disjoint. Summing over the path gives $\sum_{k=1}^{K} s(k) = O(f+1)$, and (1) yields $K = O(\sqrt{(f+1)\log(f+2)})$.

Theorem 4. *Against $f \le t < n/3$ corruptions, $\mathsf{RCE}^\star$ terminates within $f + O(\sqrt{(f+1)\log(f+2)})$ rounds.*

Proof. By Lemma 31, it suffices to bound $S_1^\star$. At each nonterminal path node of index k, the king committee contains at least $s(k)$ corruptions and its BA takes at most $s(k) + d_{\mathsf{BA}}$ rounds. For a bad left subtree rooted at v, Lemma 30 gives $D_v^\star \le |C_v \cap F| + O(N_v^\star)$. The charged sets are disjoint, so the size dependent terms sum to at most f.

Let $K := k_{u_r}$. The skipped subtrees together contain at most K nodes, and the path itself contains at most K nodes. The terminal BA has at most $3(s(K)+1)$ parties and takes $O(s(K)+1) = O(K)$ rounds, since $s(K) \le K$. Thus $S_1^\star \le f + O(K)$, and Lemma 32 gives the claimed bound.

Theorem 5 (Security of $\mathsf{RCE}^\star$). *For $t < n/3$, $\mathsf{RCE}^\star((P_1,\dots,P_n),1,r_0)$ satisfies Byzantine agreement against $f \le t$ adaptive corruptions.*

Proof. If all honest inputs are v, Lemma 23 makes every honest output equal to v, and Theorem 4 guarantees that every honest party outputs. Agreement and termination follow from Lemma 31 and Theorem 4.

Theorem 6. *The protocol $\mathsf{RCE}^\star$ communicates $O(n(t+1))$ bits.*

Proof. At a node u, the non-BA communication is $O(n_u^2)$ and the child committees satisfy $|C_u^1|^2 + |C_u^2|^2 \le n_u^2/2$. These costs therefore sum to $O(n^2)$ over the tree. Write $K := k_{u_r}$ for the terminal index. The timing argument in the preceding communication analysis also applies to this schedule, so at most $K+1$ BA instances begin before termination, each among $O(s(K+1)) = O(s(K))$ parties. Using $\mathsf{B}(m) = O(m^3\log(m+1))$, their total communication is $O(Ks(K)^3\log(s(K)+1)) = O(K^4/\log^2(K+1))$. By the bound $\sum_{k=1}^{K} s(k) = O(f+1)$ from Lemma 32 and (1), this is $O((f+1)^2)$.

More generally, a schedule $s(k) = \Theta(k/\log^\alpha(k+1))$ for fixed $\alpha \ge 0$ gives $K = O(\sqrt{f+1}\,\log^{\alpha/2}(f+2))$ by the same argument. The BA communication is then $O(K^4/\log^{3\alpha-1}(K+1)) = O((f+1)^2\log^{1-\alpha}(f+2))$. Taking $\alpha \ge 1$ makes this $O((f+1)^2)$, and $\alpha = 1$ gives the smallest terminal index bound among these choices.

Run $\mathsf{RCE}^\star$ among $3t+1$ parties and let every remaining party output after receiving the same value from $t+1$ committee members. At least $2t+1$ honest committee members send the common output, while at most t members can support another value, so every accepted value agrees and every honest party terminates. The selected committee contains at most f corruptions. Its internal $O((t+1)^2)$ non-BA cost and $O((f+1)^2)$ BA cost, together with the $O(n(t+1))$ dissemination cost, give $O(n(t+1))$ bits in total. Dissemination adds one round.

Theorem 7 (Recursive escalation instantiations). *For every execution with $f \le t$ adaptive corruptions, recursive escalation gives the following deterministic binary Byzantine agreement protocols:*

1. *for $t < n/2$, with a PKI and the stated unique threshold signatures, $f + O(\sqrt{f+1})$ rounds and $O(n(t+1)\kappa)$ bits;*

2. *for* $t < n/3$*, without cryptographic setup,* $f + O(\sqrt{f+1})$ *rounds and* $O(n(t+1) + f^2 \log(f+2))$ *bits;*
3. *for* $t < n/3$*, without cryptographic setup,* $f + O(\sqrt{(f+1)\log(f+2)})$ *rounds and* $O(n(t+1))$ *bits.*

Proof. The first two claims follow from Theorem 2 and Corollaries 3 and 4 and the preceding communication analysis with $c = 2$ and $c = 3$, respectively. The third follows from Theorems 4 to 6.