Leader Election with Poly-logarithmic Communication Per Party

Amey Bhangale[5], Chen-Da Liu-Zhang[4], Julian Loss[1,2⋆], Kartik Nayak[3], and Sravya Yandamuri[1,2,3⋆⋆]

[1] Common Prefix
[2] CISPA Helmholtz Center for Information Security
[3] Duke University spy4@duke.edu
[4] Lucerne University of Applied Sciences and Arts & Web3 Foundation
[5] University of California Riverside

Abstract. The leader election problem requires a set of n parties, out of which up to t can be Byzantine, to elect a leader uniformly at random such that no two parties disagree on the elected leader and an honest leader is elected with constant probability. The Scalable Leader Election protocol published in SODA'2006 is an important breakthrough in solving this problem efficiently for all but $o(1)$ of the parties. They achieve a protocol for $t < (\frac{1}{3} - \varepsilon)n$ (for $\varepsilon = o(1)$) in the full-information setting such that every party only sends polylog(n) bits.
In this paper, we revisit their work and show that there are subtleties in the protocol that are not dealt with in the analysis. In particular, two mechanisms related to "silencing" parties and dealing with "bad nodes" are at odds with each other, which is why the existing analysis is insufficient. We present these concerns in detail and subsequently present a modification to their protocol with a corresponding analysis to solve leader election with the desired metrics.

1 Introduction

Leader election, introduced by Ben-Or and Linial [BL85], is an important problem in distributed computing. At a high level, the leader election problem among a set of n parties out of which up to t can be Byzantine (arbitrarily malicious) requires the following properties to hold when electing a leader uniformly at random: (i) Agreement: no two honest (non-Byzantine) parties disagree on the elected leader, (ii) Validity: with constant probability, the elected leader is an honest party, and (iii) Termination: All honest parties output a leader. Leader election is an important problem in distributed computing, and it particularly finds several applications for solving related consensus problems [LSP82,CL$^+$99] (such as Byzantine Broadcast, Byzantine agreement, and state machine replication), all of which form the core underpinning of blockchains.

Owing to the requirements of modern-day blockchains, scaling consensus and, thus, correspondingly scaling leader election protocols to a large number of parties is an important line of work. One such key metric to scale is communication complexity, i.e., the number of bits sent by honest parties during a protocol execution. Through the Dolev-Reischuk [DR85] lower bound, it is known that a deterministic BA/BB incurs at least $O(t^2)$ messages. Thus, obtaining a leader election protocol with sub-quadratic communication forms an important stepping stone toward a sub-quadratic communication BA/BB protocol.

The "Scalable Leader Election" protocol [KSSV06a] published in SODA'2006 is an important breakthrough result towards achieving this goal. Particularly, they show a leader election protocol for $t < (\frac{1}{3} - \varepsilon)n$ (for some $\varepsilon = o(1)$) that incurs sub-quadratic communication complexity in such a way that every party sends messages to, and receives messages from only a polylogarithmic (in n) number of parties during an execution. In addition, the protocol has a latency polylogarithmic in n, and works in a full-information model (and thus does not rely on cryptography). Several followup works have used this result as a building block towards designing BA/BB protocols with sub-quadratic communication complexity [KSSV06a,KS10].The core idea in this work is very elegant. It involves using a layered network of *nodes* where the network is constructed using samplers. On each of the layers, elections are conducted among these nodes containing polylogarithmically many parties. Elections in each layer reduces the number of parties by a factor of $\ln n$;

⋆ This work is funded by the European Union, ERC-2023-STG, Project ID: 101116713. Views and opinions expressed are however those of the author(s) only and do not nec- essarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.
⋆⋆ Lead Author.

eventually reducing the number of parties to a polylogarithmic number out of which one party is elected as the leader. Owing to the use of samplers, their guarantees on agreement and liveness hold for all but $o(1)$ fraction of honest parties.

Our contribution. In this paper, we revisit this result and make the following key contributions:

1. **Concerns with the analysis in [KSSV06a].** We show that there are subtleties that are not dealt with by their analysis. In particular, their protocol relies on two key aspects. First, due to the use of samplers, there are a few nodes in each layer that are "bad" (w.r.t. the assumptions on the Byzantine fraction); their protocol analysis relies on bounding the number of such bad nodes. Second, their protocol relies on silencing parties if the parties are about to send too many messages in a given layer; thus, the silencing mechanism ensures that the per-party communication complexity is bounded. In our work, we show that the way to tackle bad nodes is at odds with the silencing mechanism. Without additional analysis, it is possible for a linear fraction of parties to go silent in a given layer, potentially causing the protocol to not terminate.
2. **Towards achieving a scalable leader election protocol.** The main problem with the above analysis is that, in principle, *anything* can happen in a bad node. To address the above issue, we present and provide a non-trivial modification and new analysis of their protocol and show that the desired properties can again be obtained for the modified protocol. Our modification includes the usage of graphs with improved expander properties and a novel mechanism to perform sequential election procedures with the guarantee that adversarial capabilities are highly restricted even in bad nodes, which in turn allows us to bound the impact on the remaining ones. Specifically, we obtain the following result:

Theorem 1. *Consider a network of n parties, s.t. fewer than $\frac{1}{3} - \varepsilon$ for $\varepsilon = O(\frac{1}{\ln \ln n})$ are Byzantine. There is a protocol in the full information setting that elects an honest leader with constant probability that:*

1. *incurs only* polylog(n) *bits of communication complexity per party*
2. *ensures that all but* $o(1)$ *of the parties know the leader*
3. *completes in* polylog(n) *rounds*

Further Related work. The seminal work by Ben-Or and Linial [BL85] introducing leader election lead to a significant line of work focused on improving its efficiency [AN90,BN93,ORV94,RZ98,Fei99,KSSV06a,KS10]. Similarly, several works have proposed Byzantine agreement protocols with subquadratic communication. The works [Mic17,PS17,ACD+19,GPS19] gave improved protocols with subquadratic communication complexity using the "player replaceability paradigm," which requires setup in the form of verifiable random functions. Abraham et al. [ACD+19] show a BA protocol with adaptive security and subquadratic communication complexity in the *partially synchronous* model. Cohen et al. [CKS20] show an adaptively secure asynchronous BA protocol with $o(n^2)$ communication, in a model in which the adversary cannot arbitrarily schedule delivery of messages. Blum et al.[BKLZL20] proposes a protocol for adaptively secure asynchronous BA without the assumption on message scheduling but that uses stronger computational assumptions and a trusted dealer to get subquadratic communication complexity, as well as an alternate construction that avoids those assumptions but gets only amortized subquadratic communication complexity. Recent works have presented round-efficient Byzantine Agreement protocols in the information theoretic, *asynchronous* setting [HPZ22], [HPZ24]. In [BCG21], the authors focus on *balanced* Byzantine agreement protocols, in which all parties send the same number of bits.

Technical Overview

We present an overview of the key ideas in this work. The description is split into three parts. First, we present a brief overview of the Scalable Leader Election protocol [KSSV06a] (Section 1.1). We refer to this protocol as SLE. Second, we present the key subtleties and concerns with the analysis of the protocol (Section 1.2). Finally, in Section 1.3, we present the modifications necessary to fix the protocol.

1.1 The SLE Protocol

The goal of SLE is to elect an honest leader from a distributed network of n parties with constant probability in a scalable manner without the use of cryptography. We assume that $(\frac{1}{3} - \varepsilon)n$ of the parties (for $\varepsilon = O(\frac{1}{\ln \ln n})$) in the network are faulty. Faulty parties are Byzantine, i.e., they may deviate arbitrarily from the protocol. Toward scalability, this protocol aims for a per-party communication complexity that is sublinear (in fact, polylogarithmic) in the number of parties and, thus, an overall sub-quadratic communication complexity.

Using a layered network to reduce communication complexity. In SLE, every party speaks with only poly-logarithmically many other parties. This is achieved using a layered network structure (cf. Figure 1) where, as the layers increase, the number of parties per layer decreases by a $\ln n$ factor. The topmost layer contains polylogarithmically many parties, out of which a leader can be elected. It is necessary that the number of parties decreases from layer ℓ to layer $\ell + 1$ without all of the parties on a given layer communicating with each other. Otherwise, at least at the bottom-most layers, this would incur at least quadratic communication complexity.

In the SLE protocol, each layer is made up of *nodes*, each of which contains polylogarithmically many parties. Only the parties from a specific set of nodes on a given layer may be chosen to occupy a specific node on the next layer. The parties that occupy a node are chosen from the parties in the nodes adjacent to that node on the previous layer. Both the nodes adjacent to a node on the previous layer, as well as the parties within them, are referred to as its *committee*, and the parties that are chosen to occupy the node from the committee are referred to as its *subcommittee*. As an example, in the figure, node A on layer ℓ has in its committee all the parties in $\cup_{i=1}^{5} A_i$ whereas its subcommittee consists of only parties in A (while we use set notation for simplicity, in reality the same node may appear multiple times in a committee). Looking ahead, the parties in a node's committee participate in an election, and the winners of the election occupy the node on the next layer, forming its subcommittee.

A naïve assignment of parties to nodes and creation of edges between nodes on consecutive layers can result in many nodes containing too many ($\frac{1}{3}$ or more fraction) Byzantine parties. For this reason, samplers are used to assign parties to the layer 0 nodes and to create edges between nodes on successive layers.[6] The samplers ensure the following: (1) on the bottom-most layer, all but $\frac{1}{\ln^2 n}$ fraction of the nodes contain fewer than $\frac{1}{3}$ Byzantine parties. (2) For any two layers ℓ and $\ell + 1$, let S be a set of nodes on layer ℓ that makes up s fraction of the nodes on layer ℓ; then all but $\frac{1}{\ln^2 n}$ fraction of the nodes on layer $\ell + 1$ have $s + \frac{1}{\ln n}$ fraction of their neighbors from S. This limits the power of the adversary to sabotage the protocol because it ensures that the less-than-$\frac{1}{3}$ corruption level of the entire network is preserved for most nodes and most elections. This is necessary because the election protocol requires that fewer than $\frac{1}{3}$ of the participants are Byzantine. Given this and the fact that polylogarithmically many parties participate in each election, the election participants may run a broadcast and it is possible to use a black-box election protocol to run the elections.

A depiction of the layered network and the samplers used to construct it are shown in Figure 1. On the left is the layer number followed by the number of nodes on that layer. The edges between the nodes on layers $\ell - 1$ and ℓ are assigned using the bipartite graph $G(L_\ell, R_\ell)$, for layers $\ell = 1 \ldots \ell^\star - 1$. Parties are assigned to layer 0 nodes using the graph $G(L_0, R_0)$. The bipartite graphs are samplers. The penultimate layer contains polylogarithmically many nodes, which all have edges to the single node on layer $\ell^\star$. In each bipartite $G(L_\ell, R_\ell)$, the nodes in L_ℓ have degree $\ln^4 n$, and the nodes in R_ℓ have degree $\ln^5 n$. For a node on layer ℓ, the parties in the nodes adjacent to it on layer $\ell - 1$ (its committee) participate in the election to determine who will occupy this node and form its subcommittee. As per the use of samplers, the same node (and hence, the parties within it) participate in multiple (polylogarithmically many) elections on the next layer.

Electing parties to a node. SLE uses the lightest bins protocol by Feige [Fei99] to run elections. Briefly, in this protocol, each participant chooses a "bin" among a set of bins uniformly at random, and sends their choice to all of the other parties. The lightest bin, i.e. the one that is chosen by the fewest number of parties, is the winning bin. The parties that chose that bin are determined to be the winners. Observe that due to the presence of Byzantine parties, it is necessary that the parties run *consensus* to agree on the bin choices of

[6] See Definition 3 in Section 2.2 for the formal definition of a sampler.

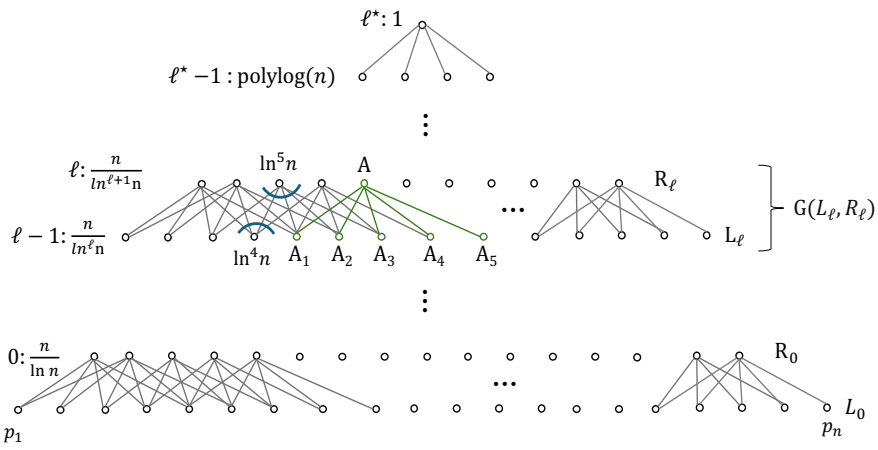

Fig. 1. The Layered Network.

the parties so that they all agree on the lightest bin (and the winning parties). Achieving consensus requires that fewer than $\frac{1}{3}$ of the parties participating in the election are Byzantine. When this condition is met, it is guaranteed that there is agreement on the bin choices of all the parties and, therefore, on the winning bin and the winning parties. The lightest bins protocol is robust because, due to the ε slack in the overall corruption threshold, the fraction of elected parties that are Byzantine remains less than $\frac{1}{3}$. Intuitively, this follows from the fact that honest parties choose their bins uniformly at random and adversarially adding Byzantine parties to a bin makes it heavier.

Communicating the leader to all parties. Together, the above approach ensures that there are only a poly-logarithmic number of parties in the penultimate layer from which a single party can be elected as a leader. The protocol relies on a communication tree to communicate this leader to all parties. The communication tree is rooted at the single node on layer $\ell^\star$, and all of the nodes in the layered network are nodes in the communication tree. The edges in the communication tree are only a specific subset of the edges in the layered network. Specifically, the i-th node on layer ℓ has as its children in the communication tree the nodes $i \ln n, \ldots, (i+1)\ln n - 1$, where the nodes on layer ℓ are numbered $0, \ldots, \frac{n}{\ln^{\ell+1} n} - 1$. The single node on layer $\ell^\star$ has as its children all the nodes on the penultimate layer. The communication tree ensures that all but $o(1)$ of the partyors know the winner of the leader election. Each time an election completes, the parties in the nodes that participated in that election pass the list of winners to all parties in their child nodes. This method of passing information to child nodes in the communication tree occurs recursively until the information is passed to the leaf nodes of the respective trees.

1.2 Subtleties and Concerns with the SLE Protocol

While the above approach is elegant and intuitively seems to work, there are several subtleties and concerns that need to be addressed. First, to participate in an election protocol, a party needs to know which parties occupy other nodes. For example, for an election to node A in the figure, parties in nodes A_1 need to know parties that won the election to a peer node, say, node A_2. Second, while the above idea seems to suggest that the communication complexity is sub-quadratic, it is not clear whether every party incurs sub-linear communication complexity. For instance, it is possible that a party may win many elections up the layered network and exceed the desired per-party communication complexity as a result. Finally, while the samplers ensure good properties for most nodes, and hence most elections, on a layer, there are $O(\frac{1}{\ln^2 n})$ fraction "bad" nodes. How are these elections handled?

Communicating the parties in peer nodes. To learn the parties in the peer nodes, SLE uses the notion of *monitoring sets* in the communication tree. Each node has a predefined monitoring set, a list of layer 0 nodes. Specifically, the monitoring set of a node is the leaf nodes of the communication trees rooted at each

node in its committee. Using the communication pattern analogous to learning the leader, the monitoring set of a node learns the list of election winners for that node. It is the job of the monitoring set of a node on layer ℓ to provide this information to the monitoring sets of appropriate peer nodes on layer ℓ so the parties in those nodes may know the appropriate set of parties in elections to nodes in layer $\ell+1$. Given that the monitoring sets of two nodes on the same layer contain the same number of layer 0 nodes, conveying this information via one-to-one communication between the nodes in the monitoring sets ensures that the necessary communication complexity is preserved. See Figure 2 for a depiction of the monitoring set of a node. Similarly to how up to $\frac{1}{\ln^2 n}$ fraction of the nodes per layer may have $\frac{1}{3}$ or more fraction Byzantine parties participating in their elections, with the use of samplers, it is also the case that up to $\frac{1}{\ln^2 n}$ nodes per layer may have many nodes in their monitoring set with many Byzantine parties. The monitoring sets of such nodes may equivocate and convey the wrong election results to the parties in other nodes. Since a party in a node learns the parties in other nodes through its own monitoring set, a party in such a node may also learn the wrong contents of other nodes. We will describe this problem in more detail as well as the implications of the lack of communicability of election results later.

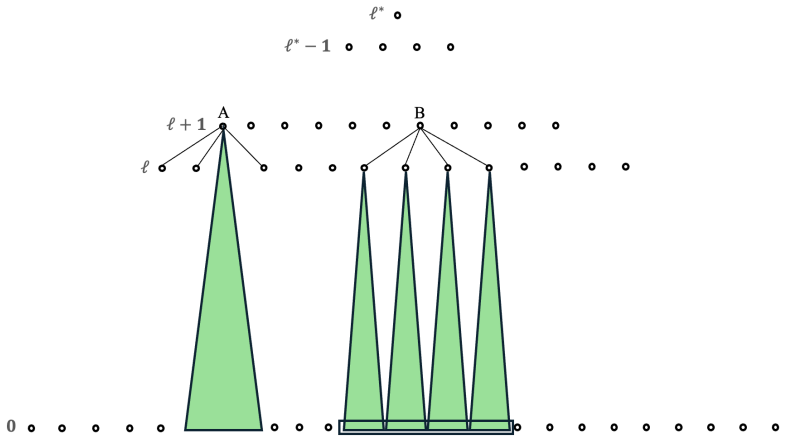

Fig. 2. Communication tree rooted at a layer $\ell + 1$ node A vs. the monitoring set of a layer $\ell + 1$ node B. The monitoring set of a node consists of the leaf nodes of the trees rooted at each node in its committee, while the children of a node in the communication tree consist of a subset of the nodes in its committee.

Ensuring sublinear communication complexity per party and tackling "bad" nodes. Recall that it is potentially possible that a party may win many elections up the layered network and exceed the desired communication complexity. In fact, without additional modifications, this does happen for at least a small fraction of the parties. SLE addresses this using a *silencing* mechanism. If a party is elected more than eight times on a given layer, the party goes silent for all nodes beyond the first eight to which it wins elections on that layer (it is not specified how exactly a party chooses which of those nodes to go silent in). Silencing is necessary from the standpoint of ensuring the desired per-party communication complexity; however, from the perspective of a given node, silencing an honest party implies the presence of one fewer honest party in that election. If a node contains many silent parties, the non-silent parties could contain a majority of Byzantine parties. Thus, it is necessary to show that too many honest parties do not go silent on a given layer.

In the analysis of SLE, [KSSV06a, Lemma 5.1] claims that if X is the total number of parties on layer ℓ of the network (counting multiplicities), then at most $O(X/\ln^3 n)$ parties are silent on the next layer (counting multiplicities). This bound is sufficiently small so that the silent parties in (most of) the nodes can be absorbed in the ε slack in the Byzantine fraction of the network overall (recall that the protocol tolerates $(\frac{1}{3}-\varepsilon)n$ Byzantine parties). So, the honest-Byzantine ratio in a vast majority of the nodes stays at the desirable threshold. However, while the idea intuitively makes sense, it turns out that there is a subtle flaw in the proof, which is why the statement does not hold.

This flaw is related to the third concern on how bad nodes are tackled, so let us first reason about it. Recall that the use of samplers implies that there are $O(\frac{1}{\ln^2 n})$ fraction of bad nodes in a given layer. In such nodes, the elections may contain $\frac{1}{3}$ or more fraction Byzantine parties. The protocol analysis assumes that elections in such nodes still essentially work despite the protocol assumption on the Byzantine fraction not holding true. From [KSSV06a, Lemma 5.1]: "Being overly generous, suppose that for each election the adversary is able to choose which bin is elected." The adversary can certainly choose the winning bin in such an election; however, when using a black-box agreement protocol if the assumption on Byzantine fraction is not satisfied, the agreement property may not hold either. In effect, the adversary may choose to have multiple participating parties think that theirs is the winning bin, even if they all chose different bins.

We proceed to describe the concerns with silencing and bad nodes in more detail.

Approach to bad elections and silencing in [KSSV06a, Lemma 5.1]. The approach to proving the bound on the number of silent parties on each layer is as follows:

1. Assuming that a party is silent for all but 8 nodes on layer ℓ, they only participate in elections to nodes adjacent to those eight nodes on layer $\ell + 1$.
2. For each of those elections it participates in, the party chooses a bin uniformly at random in the lightest bins procedure.
3. In the worst case, the adversary can choose the single winning bin for each election, and only parties that chose that bin believe that they are elected.
4. Given that there may only be one winning bin per election, and honest parties choose their bin for each election uniformly at random, the number of honest parties that win eight or more elections on the next layer is very low, even with adversarially chosen winning bins.

The flaw in this approach is in step 3. Because the layered network is created using samplers, and parties are assigned to nodes on layer 0 of the network using samplers, there are elections to which more than $\frac{1}{3}$ fraction of the parties participating are Byzantine.

For the elections in which this is the case, there is no consensus, and hence there is *no agreement nor validity on the bin choices of the parties*. This is the case even for the bin choices of the honest parties. For instance, if honest party p_1 chooses bin 5, honest party p_2 participating in the same election protocol may learn that p_1 chose bin $b \neq 5$. Another honest party p_3 may learn that p_1 chose bin $b' \neq b \neq 5$. The result is that each honest party participating in the election may have a completely different view of the bin choices of all the parties. They may therefore each think that the bin they chose is the winning bin. As a result, it did not matter that each honest party participating in the election chose a bin uniformly at random; the adversary did not have to choose a single winning bin, and all of the parties thought the bin they chose was the winning bin. For the rest of this section, we will refer to this phenomenon as *multiple-winning-bins*.

Potential shadow elections in bad nodes. If this problem occurs for the election to a node on layer ℓ, a natural question is whether a party could notice something had gone wrong during the elections to layer $\ell + 1$; however, the potential lack of communicability of election results hampers this possibility. Recall that in order to participate in the election protocol, parties need to learn which parties occupy the other nodes in the committee. This requires that the election results are reliably communicable, but we cannot ensure this for all nodes. The reason for this is that this information is learned using the monitoring sets. For parties in node A_1 (i.e. those in its subcommittee) to learn what parties are in node A_2, A_2's monitoring set needs to convey the correct information to A_1's monitoring set. After this, the monitoring set of A_1 needs to convey the correct information to the parties in A_1. So for parties in a layer ℓ node A_1 that is in the committee of layer $\ell + 1$ node A to learn the other parties in the committee for the purposes of running the election for A, it requires that the monitoring sets of both nodes are mostly made up of nodes with a majority of honest parties. When the monitoring set of A_1 is not made up of many such nodes, and therefore may not reliably communicate information, a party in A_1 can learn the wrong list of parties in the committee of A_2. The adversary can convince this party that all of the other parties in the committee are a wrong list of Byzantine parties. In this case, the party participates in what we refer to as a *shadow election* to node A. For the remainder of this section, we will refer to nodes whose monitoring sets do not reliably convey information as nodes with an *equivocating monitoring set*.

A shadow election is an election to a node that occurs in parallel to the *actual* election to that node. The actual election is the one run among the honest parties in the nodes *without* an equivocating monitoring set in the committee of that node. In the shadow elections run for the same node, the adversary is able to tailor a view of the election particularly for each party because the monitoring set can convey to each party a separate and wrong list of parties in the other nodes in the committee. Since the adversary has full control over shadow elections, and the committees in which they are run have no bound on the Byzantine fraction, the adversary can ensure that every honest party believes they won every shadow election they participate in. The result is that, once an honest party is elected to a single node with the multiple-winning-bins problem, they are convinced by the adversary that they won this election. If this node also has an equivocating monitoring set, they may also be convinced that they win every election in every node adjacent to this node on the next layer.

One question then is whether a party that believes wrongly that it was elected to a node whose monitoring set *is* able to convey the correct information, could learn that it was in fact not elected to that node. It is true that a party could conceivably learn that an election it thought it won was actually a shadow election from a monitoring set that is able to reliably convey information. But for all shadow elections it wins to nodes with equivocating monitoring sets, this party may still be convinced it's elected. And as long as it is elected to eight such nodes, it goes silent for all other nodes on that layer. Unfortunately, the SLE analysis does not separately bound the number of nodes with an equivocating monitoring set so that it is sufficiently low; such a bound is essential to argue that the above concern does not arise.

The end result of the combination of the silencing mechanism with the phenomenon of shadow elections is that all of the honest parties may cease participation in the protocol within the first few layers, because they are only *not* silent for the shadow elections that they win. Since Byzantine parties can defy the protocol and choose not to go silent, only Byzantine parties may be elected to the topmost node or the protocol may simply not terminate. [7]

Potential simple fixes that do not work. We discuss potential fixes that do not work. One approach is to increase the silencing threshold from eight elections to some $O(\ln^k n)$ elections; however, this would not help. First, this would increase the total number of elections that a party participates in on the next layer, causing the number of legitimate silent parties on the next layer to increase. Secondly, increasing the silencing bound may reduce the number of silenced parties from one layer to the next, but with the shadow election phenomenon and the fact that a node is in the committee of polylogarithmically many nodes on the next layer, it may result in the same number of silenced parties overall.

A second approach may be to simply change the sampler properties to reduce the fraction of such nodes on each layer from $\frac{1}{\ln^2 n}$ to a smaller fraction; however, this would not help either, as changing the sampler properties in such a way would necessitate increasing the degrees of the bipartite graphs. So while the fraction of "bad" nodes on a given layer may decrease, the number of nodes in its committee, and therefore the number of parties affected by it, would increase, rendering the fix ineffective.

Corrigendum to [KSSV06a]. The authors of [KSSV06a] have added a corrigendum to address the problem with SLE. It references ([KSSV06b]), which presents a protocol (which we refer to as P2P) that solves the scalable leader election problem in a peer to peer network with a degree that is polylogarithmic in its size. The authors claim that the solution to P2P addresses the problem with SLE. Like SLE, P2P uses a silencing mechanism to ensure load-balancing. If a party wins more than 8 elections on a given layer, it goes silent for the remaining elections that it wins on that layer, as in SLE. To show that this does not cause too many honest parties to go silent on each layer, [KSSV06b, Lemma 7.2] contains a claim similar to that of [KSSV06a, Lemma 5.1]. However, the paper does not include a proof for this lemma, and only claims that [KSSV06a] contains a proof to a similar lemma, which would be [KSSV06a, Lemma 5.1]. As shown earlier in this technical overview, the proof of this lemma contains a crucial flaw because it fails to address the multiple-winning-bins problem. Without a full, standalone proof of P2P we are unable to verify the correctness of P2P and whether this work does in fact fix the problems with the SLE protocol.

[7] Note that what we have shown is an issue with the proofs; it does not imply a concrete attack, while one may exist.

1.3 Fixing the SLE Protocol

Consider the hypothetical scenario in which, for each node, there exists an agreed upon list of parties that could learn the election results, filter them so that there is only a single winning bin, and pass the results back to the committee, and that it could do so using only polylogarithmic communication complexity per party. If this existed, the parties participating in an election could consult this group of parties after running the lightest bins protocol, and it would ensure that the assumption that there is a single winning bin for every election holds. A candidate for this scenario is the monitoring set of a node. However, as mentioned previously, we are unable to derive a bound on the fraction of nodes per layer with equivocating monitoring sets. Perhaps instead, every node could have a predefined *group* of monitoring sets which could do this filtering of election results. While the bound on the number of nodes with an equivocating monitoring set may be high, the number of nodes whose *groups of monitoring sets contain many equivocating monitoring sets* may be much lower. We use this intuition to arrive at our fix, which we describe next.

Our fix to the protocol can be divided into two main components. First, we modify the samplers that are used to construct the layered network so that they are also *good expanders*. While the original samplers ensured that the layer ℓ nodes adjacent to most layer $\ell+1$ nodes comprise a "representative sample" of the layer ℓ nodes, they did not make any such guarantees in the opposite direction. To see why this addition is helpful, we come to the second part of our fix.

The second component of our fix is to introduce a new procedure to be run after the lightest bins protocol to confirm to parties whether or not they won an election. This procedure, ConfirmElectionWinner, uses specific monitoring sets of nodes 3 layers above an election node to perform this confirmation. Let $adj^+(A)$ be the nodes adjacent to layer ℓ node A on layer $\ell+1$, and let $adj_2^+(A)$ refer to $adj^+(adj^+(A))$, and so on. We use the monitoring sets of the nodes in $adj_3^+(A)$ for confirmation of the winners to A. The key is that the fraction of nodes A per layer that have many monitoring sets of the nodes in $adj_3^+(A)$ containing many layer 0 nodes with more than $\frac{1}{3}$ Byzantine parties is far lower than the original fraction of problematic nodes per layer that were given using samplers. This is due to the additional expander properties of the bipartite graphs that are used to construct the layered network. The election winner confirmation procedure ensures that even if a node A has more than $\frac{1}{3}$ Byzantine parties participating in its election, the adversary is forced to choose a single winning bin. The fraction of nodes per layer that are resistant to this procedure is sufficiently small, and as a result we are able to derive a bound on the fraction of silent parties per layer that does not inhibit completion of the protocol. We show that this procedure limits the power of the adversary in most elections per layer using the expansion properties of the network. They key to this fix is that it sufficiently reduces the number of "problematic" nodes in the network *without* a commensurate increase in the degrees of the network (and therefore the number of parties impacted by those nodes).

To use this approach, we must overcome certain challenges. One such challenge comes from the reliance on the monitoring sets to send messages to the election winners. Since monitoring sets are not of polylogarithmic size for all layers of the network, a naïve approach may result in violating our desired communication complexity bound. To overcome this challenge, we introduce a polling mechanism so that parties never receive messages from an entire monitoring set, but they are still guaranteed to learn the correct information with high probability. Another challenge is related to the protocol analysis. As described, the adversary may choose to silence all of the honest parties participating in a layer ℓ bad election via the silencing mechanism at once on layer ℓ. On the other hand, the adversary could choose not to immediately silence these parties, and silence many such parties on some higher layer via shadow elections[8]. Our analysis must show that even if the adversary takes this second approach, collecting many honest parties layer by layer that it has the power to silence and choosing to silence them at once at some higher layer, the total number of honest parties that may be silenced on any given layer is sufficiently low to ensure that the protocol completes. In Section 6.1, we combine the additional expansion properties of our layered network, the ConfirmElectionWinner procedure for election winner confirmation, and a robust worse-case analysis of adversarial behavior to present a revised, corrected version of [KSSV06a, Lemma 5.1].

Our solution preserves the sub-quadratic communication complexity per-party as follows. For the election to a node A, the additional communication for ConfirmElectionWinner consists of communication of election winners from the monitoring set of each node in A's committee to the monitoring sets of the nodes in $adj_3^+(A)$. The size of the monitoring sets of the nodes increase by a polylogarithmic factor for each layer up the

[8] See Section 6.1 for a more detailed description of this problem, which we refer to as *latently silent honest parties*.

network. Because of this, the nodes in the monitoring sets of nodes in A's committee each communicate with polylogarithmically many nodes in the monitoring sets of nodes in $adj_3^+(A)$. The parties in the monitoring sets of the nodes in $adj_3^+(A)$ determine the election winners based on the information they received, and send a confirmation message *only* to the (at most) $\ln^8 n$ winners of the election from which they have received polling messages. A party that wishes to confirm whether it has won an election chooses polylogarithmically many nodes to poll. As in SLE, we use a silencing mechanism, so that parties go silent for all but 10 nodes to which they win elections on a given layer. While we increase degrees of the network to ensure that the samplers are also good expanders, we maintain that the degrees are still polylogarithmic. This ensures that a party is in polylogarithmically many monitoring sets per layer, preserving the sub-quadratic per-party communication complexity.

1.4 Organization

The rest of the paper is organized as follows. In Section 2, we present the model and preliminaries. This includes a description of the bipartite graphs that we use, along with proofs related to their sampler and expander properties. In Section 3, we present an overview of the entire supreme committee election protocol, including details on the construction of the layered network. In Section 4, we present the subcommittee election protocol in detail. Then in Section 5, we present our addition to the subcommittee election protocol, in the form of ConfirmElectionWinner, which address the concerns raised with SLE. Finally, in Section 6, we analyze the full protocol. First, we show a bound on the number of silenced honest parties per layer. Finally, we analyze the full protocol, culminating in the proof of Theorem 1.

2 Model and Preliminaries

We assume a network $\mathcal{P}$ of n parties, where each party is connected via a point-to-point authenticated channel. We assume static corruptions. The adversary may corrupt $t < (\frac{1}{3} - \varepsilon)n$ parties prior to the start of the protocol, where $\varepsilon = O(\frac{1}{\ln \ln n})$. Corrupted parties are Byzantine, meaning that they may arbitrarily deviate from the protocol.

Our protocol works in the information theoretic model, meaning that we do not rely on any cryptographic assumptions. We assume a *synchronous* network, in which messages are guaranteed to arrive by some delay Δ. We say that an event occurs with high probability (w.h.p.) if it happens with probability at least $1 - n^{-\omega(1)}$.

2.1 Important Problems

Our protocol is designed to solve *leader election*, which is defined as follows.

Definition 1 (Leader Election). *Let Π be a protocol executed by a set of parties $\mathcal{P} = \{p_1 \dots p_n\}$, where each party p_i outputs an index of a party $p_i^\star \in \mathcal{P}$. Π is a Leader Election protocol if the following properties hold whenever at most $t < (\frac{1}{3} - \varepsilon)n$ parties are corrupted:*

- ***Validity:*** *With constant probability, $p_i^\star$ is the index of an honest party.*
- ***Agreement:*** *For all but $o(1)$ fraction of the honest parties p_i and p_j, $p_i^\star = p_j^\star$.*
- ***Liveness:*** *All but at most $o(1)$ honest parties output a value.*

2.2 Samplers

In order to achieve polylogarithmic per process communication complexity, we follow the approach of [KSSV06a] in constructing a layered network using a series of bipartite graphs that are both samplers and good expanders.

We now present the lemma describing the sampler used in our work. We also need a stronger expansion property (property 5 below) in our layered network for our proof.

Lemma 1. *There exists a set of bipartite graphs $G(L_i, R_i)$ for $i = 0 \dots \ell^\star$ such that $\ell^\star$ is the maximal integer such that $\frac{n}{\ln^{5\ell^\star} n} \leq \ln^{20} n$ and for all i, $|L_i| = n/\ln^{5i} n$ and $|R_i| = n/\ln^{5i+5} n$. In addition, for all $i = 0 \dots \ell^\star$ the bipartite graphs have the following properties:*

1. *The degree of each node in L_i is $\ln^6 n$.*
2. *The degree of each node in R_i is $\ln^{11} n$.*
3. *For any subset L'_i of L_i, at most $\frac{400}{\ln^2 n}$ fraction of the nodes in R_i have more than $\frac{|L'_i|}{L_i} + \frac{1}{\ln^2 n}$ fraction of their neighbors from L'_i.*
4. *Label each node in R_i as $0 \dots |R_i|$ and each node in L_i as $0 \dots |L_i|$. The k-th node in R_i is incident to nodes $k \cdot \ln^5(n) \dots (k+1) \cdot \ln^5(n) - 1$ in L_i.*
5. *(Expander-mixing property) For every subsets $S \subseteq L_i$ and $T \subseteq R_i$, we have*

$$\left| e(S,T) - \frac{|S||T|d_L}{|R_i|} \right| \leq \lambda \sqrt{|S||T|} + \max\{|S|, |T|\},$$

where $e(S,T)$ is the number of edges between S and T in G, $d_L = \ln^6 n$ and $\lambda = 10 \ln^{5.5} n$.

Biregular Expander Graphs Consider a bipartite graph on L and R with the left degree d_L and the right degree d_R.

Definition 2. *A biregular graph $G(L,R,E)$ with the left degree d_L and the right degree d_R is called a λ-expander if all of the eigenvalues of its adjacency matrix A_G except one have absolute value at most λ.*

Brito, Dumitriu, and Kameron [BDH22] showed that a random biregular bipartite graph is a good expander.

Lemma 2 ([BDH22]). *There exists a randomized construction of λ-expanders with $\lambda \leq 10\sqrt{\max\{d_L, d_R\}}$.*

A useful property of expander graphs is the so-called Expander Mixing Lemma. Roughly, this lemma states that the number of edges between two subsets of vertices in an expander graph is what is expected in a random graph, up to an additive error that depends on the second eigenvalue. In the case of a bipartite graph, the above condition holds for any two subsets, one from each side of the bipartite graph.

Lemma 3 (Expander-mixing lemma [Hae94]). *Fix a λ-expander biregular graph $G(L,R,E)$ with the left degree d_L and the right degree d_R, then for every subsets $S \subseteq L$ and $T \subseteq R$, we have*

$$\left| e(S,T) - \frac{|S||T|d_L}{|R|} \right| \leq \lambda \sqrt{|S||T|},$$

where $e(S,T)$ is the number of edges between S and T in G.

Property 3 from Lemma 1 is called a sampler property of a graph, and we define the general property here.

Definition 3. *A graph $G(L,R,E)$ is called a (θ,ω)-sampler if for every $X \subseteq L$, the reaction of $i \in R$ such that $\frac{|\Gamma(i) \cap X|}{d_R} > \frac{|X|}{|L|} + \theta$ is at most ω.*

The next lemma shows that a good expander is also a good sampler.

Lemma 4. *Every λ-expander biregular graph $G(L,R,E)$ with the left degree d_L and the right degree d_R, is a (θ,ω) sampler for any θ and ω satisfying the following condition*

$$\omega \geq \left(\frac{\lambda}{\theta d_R}\right)^2 \frac{|L|}{|R|}.$$

Proof. Fix a λ-expander biregular graph $G(L,R,E)$. Fix an arbitrary subset $X \subseteq L$. Suppose T is the set of vertices $i \in R$ such that $\frac{|\Gamma(i) \cap X|}{d_R} > \frac{|X|}{|L|} + \theta$, where $\Gamma(i) \subseteq L$ are the neighbors of i in L. By the definition of T, we have

$$e(X,T) \geq |T| \left(\frac{|X|}{|L|} + \theta\right) d_R.$$

Using the expander mixing lemma, we have

$$e(X,T) \leq \frac{|X||T|d_L}{|R|} + \lambda\sqrt{|X||T|}.$$

Combining the above two, we get

$$|T|\left(\frac{|X|}{|L|} + \theta\right) d_R \leq \frac{|X||T|d_L}{|R|} + \lambda\sqrt{|X||T|}.$$

Dividing both the sides by $|T|d_R$,

$$\left(\frac{|X|}{|L|} + \theta\right) \leq \frac{|X|d_L}{|R|d_R} + \frac{\lambda}{d_R}\sqrt{\frac{|X|}{|T|}}.$$

Simplifying this above using the fact that $|R|d_R = |L|d_L$, we get

$$\left(\frac{|X|}{|L|} + \theta\right) \leq \frac{|X|}{|L|} + \frac{\lambda}{d_R}\sqrt{\frac{|X|}{|T|}},$$

and hence

$$\theta \leq \frac{\lambda}{d_R}\sqrt{\frac{|X|}{|T|}}.$$

Finally, we get

$$\frac{|T|}{|R|} \leq \left(\frac{\lambda}{\theta d_R}\right)^2 \frac{|X|}{|R|} \leq \left(\frac{\lambda}{\theta d_R}\right)^2 \frac{|L|}{|R|}.$$

Therefore, this shows that at most ω fraction of vertices $i \in R$ are such that $\frac{|\Gamma(i)\cap X|}{d_R} > \frac{|X|}{|L|} + \theta$.

Proof (Proof of Lemma 1). Take an expander $G'(L_i, R_i, E')$ between L_i and R_i given by Lemma 2 with $d'_L = \ln^6 n - 1$ and $d'_R = \ln^{11} n - \ln^5 n$ and $\lambda = 10\ln^{5.5} n$. By Lemma 4, G' is a $(\frac{1}{2\ln^2 n}, \frac{400}{\ln^2 n})$ sampler.

Update the graph G' to $G(L_i, R_i, E)$ as follows. For a vertex labelled k in R_i add additional edges to nodes labelled $k \cdot \ln^5(n) \ldots (k+1)\cdot \ln^5(n) - 1$ in L_i. Denote the new graph by G. The left degree increase by 1 in G compared to G' which implies $d_L = \ln^6 n$. The new right degree of G is $d_R = d'_R + \ln^5 n = \ln^{11} n$ as required. The graph G satisfies properties 1, 2, and 4 from the lemma.

These additional edges will not affect the sampler and expansion properties much, as shown below.

Sampler property. For every set $X \subseteq L_i$ and $v \in R_i$ with $\frac{|\Gamma_{G'}(v)\cap X|}{d'_R} < \frac{|X|}{|L_i|} + \frac{1}{2\ln^2 n}$, we have

$$\begin{aligned}
\frac{|\Gamma_G(v)\cap X|}{d_R} &\leq \frac{|\Gamma_{G'}(v)\cap X| + \ln^5 n}{d_R} \\
&\leq \frac{|X|}{|L_i|}\frac{d_R}{d_{R'}} + \frac{1}{2\ln^2 n}\frac{d_R}{d_{R'}} + \frac{\ln^5 n}{d_R} \\
&\leq \frac{|X|}{|L_i|}\left(1 + \frac{1}{\ln^4 n}\right) + \frac{1}{2\ln^2 n}\left(1 + \frac{1}{\ln^4 n}\right) + \frac{\ln^5 n}{\ln^{11} n} \\
&\leq \frac{|X|}{|L_i|} + \frac{1}{\ln^2 n},
\end{aligned}$$

where we used the fact that $\frac{d_R}{d'_R} \leq \left(1 + \frac{1}{\ln^4 n}\right)$.

Therefore if for a given set $X \subseteq L_i$ at most an $\frac{400}{\ln^2 n}$ fraction of all vertices $v \in R_i$ have $\frac{|\Gamma_{G'}(v)\cap X|}{d'_R} > \frac{|X|}{|L_i|} + \frac{1}{2\ln^2 n}$, then at most an $\frac{400}{\ln^2 n}$ fraction of all vertices $v \in R_i$ have $\frac{|\Gamma_G(v)\cap X|}{d_R} > \frac{|X|}{|L_i|} + \frac{1}{\ln^2 n}$. This proves property 3 from the lemma for the graph G.

Expander-mixing property. For any $S \subseteq L_i, T \subseteq R_i$, the additional number of edges added between S and T in G (compared to the edges in G') is at most $\max\{|S|, |T|\}$. Thus, for the updated graph G, we have,

$$\left|e(S,T) - \frac{|S||T|d_L}{|R|}\right| \leq \lambda\sqrt{|S||T|} + \max\{|S|, |T|\}.$$

This proves property 5 from the lemma for the graph G.

3 Protocol Overview

In this section, we present an overview of the protocol. We start with the setup for the protocol: the construction of the layered network in Section 3.1. This is followed by a description of the communication tree Section 3.2. Finally, we describe the way in which parties are elected to nodes layer by layer up the tree in Section 3.3.

3.1 The Layered Network

We present the pseudocode for the construction of the layered network in Figure 4.

The layered network construction starts with a series of ℓ^* bipartite graphs that satisfy the properties of Lemma 1. The end result of this procedure is a layered network with $\ell^* + 1$ layers. There are edges between every two consecutive layers, and layer 0 nodes have parties assigned to them.

First, $n/\ln^5 n$ layer 0 nodes are created, and parties are assigned to those nodes based on the graph $G(L_0, R_0)$. Figure 3 contains a depiction of such a bipartite graph. For each edge in $G(L_0, R_0)$ between node j in L_0 and node k in R_0, party p_j is assigned to the kth layer 0 node. We specify the assignment in this way because a party may be assigned to the same node multiple times. Ultimately, each layer 0 is assigned $\ln^{11} n$ parties, counting multiplicities.

The nodes and edges in the remainder of the layered network are constructed in the same fashion. For layers $\ell < \ell^*$, $n/\ln^{5\ell+5} n$ nodes are created, and an edge is created between the j-th node on layer $\ell - 1$ and the k-th node on layer ℓ for each edge between the j-th node in L_ℓ and the k-th node in R_ℓ in $G(L_\ell, R_\ell)$.

A single node is created for layer ℓ^*, and it is connected to every node on layer $\ell^* - 1$.

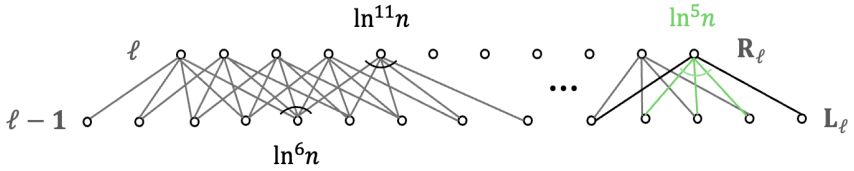

Fig. 3. The bipartite graph $G(L_\ell, R_\ell)$ used to create edges between nodes on layers $\ell - 1$ and ℓ. The nodes of R_ℓ correspond to the layer ℓ nodes and the nodes of L_ℓ correspond to the layer $\ell - 1$ nodes. For layer 0, L_0 corresponds to the n parties, and $G(L_0, R_0)$ is used to assign the parties to the layer 0 nodes.

Input: Bipartite graphs $G(L_i, R_i)$ for $i = 0, \ldots, \ell^*$
Create $n/\ln^5 n$ nodes for layer 0
for each edge in $G(L_0, R_0)$ **do**
  if the edge is between the jth node in L_0 and the kth node in R_0 **then**
    Assign party p_j to the kth node on layer 0
for $\ell = 1, \ldots, \ell^*$ **do**
  if $\ell < \ell^*$ **then**
    Create $n/\ln^{5\ell+5} n$ nodes for layer ℓ
    for each edge in $G(L_\ell, R_\ell)$ **do**
      if the edge is between the jth node in L_ℓ and the kth node in R_ℓ **then**
        Connect the jth node on layer $\ell - 1$ and the kth node on layer ℓ
  else
    Create one node for layer ℓ^* and connect it to each node on layer $\ell^* - 1$

Fig. 4. BuildLayeredNetwork

3.2 The Communication Tree and Monitoring Sets

The communication tree consists of all of the nodes in the original layered network, while the edges in the tree are only a subset of the edges in the original layered network. For the j-th node on layer ℓ of the network for $0 < \ell < \ell^*$, its children in the communication tree are nodes $j \cdot \ln^5 n, \ldots, (j+1) \cdot \ln^5 n - 1$ on layer $\ell - 1$. Note that these nodes are already connected in the layered network due to property 4 of the bipartite graphs from Lemma 1. The children of the single node on layer ℓ^* are all of the nodes on layer $\ell^* - 1$. The leaf nodes of the communication tree are the layer 0 nodes.

Monitoring Sets. Each node in the network is assigned a monitoring set whose purpose is to maintain the information of which parties are in that node, and to convey this information to parties that wish to learn it. The monitoring set of a node is a specific set of layer 0 nodes. We define a monitoring set as follows.

Definition 4 (Monitoring Set). *The monitoring set of node A, $ms(A)$ on layer ℓ for $0 < \ell < \ell^*$ is the leaf nodes of the trees rooted at each node adjacent to it on layer $\ell - 1$.*

In Figure 2, one can find depiction of the communication tree rooted at a node A and the monitoring set of a node B for two nodes A and B on the same layer. The monitoring set of the single node on layer ℓ^* is all of the layer 0 nodes. We use the notation $ms(A)$ to refer to the monitoring set of node A.

3.3 Supreme Committee Election

Now that we have shown how the layered network is constructed and how the communication tree and monitoring sets are defined, we are ready for the supreme committee election protocol. Recall that during the layered network setup, parties are assigned to the layer 0 nodes. During the supreme committee election protocol, parties are elected to nodes layer by layer up the tree. The parties participating in the election to a given node are the parties in the nodes adjacent to it on the previous layer. For this purpose, we will use the following terminology.

Definition 5 (Subcommittee). *The subcommittee of a node is the parties within that node.*

We refer to a the parties within a node, as well as the subcommittee of a node, interchangeably.

Definition 6 (Committee). *The committee of a node is the nodes adjacent to that node on the previous layer, as well as the parties within those nodes.*

The parties in a node's commitee participate in an election, and the election winners become the subcommittee of that node. [9] Observe that in order to participate in an election to a node, a party must learn the other parties with whom to run the election protocol. This is because only the parties in the commitee of a node participate in the election to that node, and thus know the results of the election to that node. This is where the monitoring sets are used. The purpose of the monitoring set of a node is to keep track of which parties are in that node. It is the responsibility of the monitoring set to communicate this information to the monitoring sets of other nodes so that the parties in those nodes may learn this information. The monitoring set of a node learns the winners of the election to that node through SendMsgDownTree (Figure 5). The list of election winners to node A are sent from the nodes in A's commitee to the nodes in A's monitoring set, which are also their layer 0 descendants in the communication tree. To do this, SendMsgDownTree, is invoked recursively on the down the trees until reaching the layer 0 nodes. A party sends the message m to all of the parties in its child nodes in the communication tree.

[9] When we describe the protocol in this section, we assume that a party acts as a separate entity for each "slot" it occupies in a given node. That is, a party may win the election for (or in the case of layer 0, be assigned to) the same node multiple times, in which case it acts as a separate entity for each of those times that it wins, even to the same node. For example, if an instruction says that all parties in a node A do x, if a party p_i has been elected to A multiple times, or has been assigned to it multiple times in the case of layer 0, then p_i does x twice, and it will have two different states associated being in A. When describing the election protocol in further detail in subsequent sections, we are more precise on how we consider the slots in a node and how the parties occupying them behave.

Input: Each party p_j in A has input a message m_j
1: **if** $\ell > 0$ **then**
2: Each party p_j in A sends m_j to each party in the nodes $i \cdot \ln^5 n, i \cdot \ln^5 n + 1, \ldots, (i+1) \cdot \ln^5 n - 1$ on layer $\ell - 1$
3: **for** each child node B of A **do**
4: For each party p_k in B, let m_k be the message received from a majority of the parties in A and *null* otherwise
5: Invoke SendMsgDownTree on each node $i \cdot \ln^5 n, i \cdot \ln^5 n + 1, \ldots, (i+1) \cdot \ln^5 n - 1$ in layer $l - 1$, where party p_k has input message m_k

Fig. 5. SendMsgDownTree for node A, the ith node on layer ℓ

Input: Each party in each node in $ms(A)$ has a message m containing the parties in A
1: Every party in the ith node in $ms(A)$ sends m to every party in the ith node of $ms(B)$
2: A party in $ms(B)$ considers the parties in A to be the list it received from a majority of the parties in the corresponding node in $ms(A)$
3: Each party in B selects a set of $\ln^8 n$ nodes in $ms(B)$ uniformly at random and sends to all parties in those nodes a message $\langle$learn processors, $A\rangle$
4: **for** each party in $ms(B)$ **do**
5: let *poll_list* be the parties in B from which it received $\langle$learn processors, $A\rangle$
6: **for** each party p_j in *poll_list* **do**
7: send m to p_j
8: **for** each party in B **do**
9: consider the parties in A to be the list received from a majority of the polled parties

Fig. 6. LearnProcessors in which party p_i in node B wishes to learn the parties in node A

Once the monitoring set of a node knows the parties that won the election to that node, the nodes in the monitoring set are able to communicate this information to other monitoring sets that need to learn it. This communication is done via LearnProcessors (Figure 6). In LearnProcessors, if the parties in node B wish to learn the parties in node A, first every party in the i-th node of the monitoring set of node A on layer ℓ tells every party in the i-th node of the monitoring set of node B on layer ℓ the winners of the election to A. Then every party in B selects $\ln^8 n$ parties in $ms(B)$ to poll. Each party in $ms(B)$ responds to the parties in B that have polled it. Because a party in $ms(B)$ already has an idea of which parties are in B, this ensures that it only responds to polylogarithmically many parties that are polling it.

In Figure 7, we present the full election protocol, SupremeCommitteeElection. The protocol starts with the construction of the layered network. Then, the elections occur for each node, layer by layer. Prior to the elections, LearnProcessors is invoked to ensure that the parties in the committee of a node know which parties to communicate with for the election. After the election procedure, SendMsgDownTree is invoked so that the monitoring set of a node knows the election winners.

In Section 4, we present the election protocol, SubcommitteeElection. We start with the base of the election protocol: the lightest bins protocol, LightestBins. SubcommitteeElection uses an additional sub-protocol, ConfirmElectionWinner, to fix the problem in the original protocol. We present the full election protocol SubcommitteeElection in Section 4.4, which combines LightestBins and ConfirmElectionWinner. We describe ConfirmElectionWinner in more detail in Section 5 after discussing the problem with the original protocol in detail and motivating our solution.

4 Elections

In this section, we present the election protocol which is used to populate the nodes in the layered network.

To begin, each node starts with $\ln^8 n$ empty slots. These slots are filled through an election, and the parties that win the election occupy the slots and make up that node's *subcommittee* (Definition 5). The participants in the election to the node are the parties occupying the slots in the nodes adjacent to the election node on the previous layer, and we call these nodes (and the parties within them) on the previous layer the node's *committee* (Definition 6).

Input: Set of bipartite graphs $G(L_\ell, R_\ell)$ for $\ell = 0, \ldots, \ell^\star$, Set N of n parties
1: Invoke BuildLayeredNetwork on inputs $G(L_\ell, R_\ell)$, N for $\ell = 0, \ldots, \ell^\star$
2: **for** $\ell = 1, \ldots, \ell^\star$ **do**
3: **if** $\ell < \ell^\star - 4$ **then**
4: A party goes silent for all slots that it is elected to on layer ℓ after the first 10
5: **if** $\ell > 1$ **then**
6: Invoke LearnProcessors on every pair of layer $\ell - 1$ nodes adjacent to the same layer ℓ node
7: Invoke SubcommitteeElection for each layer ℓ node on the parties that occupy its committee and let W, aux be the output

Fig. 7. SupremeCommitteeElection

For the success of the overall protocol, certain properties need to be satisfied by the election protocol. First, in order for any election protocol to be executed, the parties in each node in the committee must learn which parties occupy the slots in the other nodes in the committee. Since nodes are populated through elections, this implies that the results of the election protocols are communicable in a reliable way so that the parties in other nodes may learn the correct results. A second necessary property is related to the actual election results. It must be the case that sufficiently many honest parties are elected and that the committee (at least most of the honest parties within it) agree on this outcome.

Parties in each node of the committee learn the occupants of every other node in the committee using LearnProcessors discussed in Section 3.3. Each party p_i maintains an array σ_i that indicates which party it believes occupies each slot of the committee. In the good case, most of the slots in the committee are occupied by honest parties that agree on the occupants of nearly all of the slots in the committee. To be precise, if this is the case, we say that the committee contains a *core set*, which we define in Definition 7. We show in this section that if there exists a core set, then the election protocol is successful: there is a random set of sufficiently many honest parties elected, and most of the honest parties participating in the election (those in the core set) agree on the election winners. Furthermore, when the node to which the election is occurring is to a specific class of nodes that make up most nodes of the network, which we refer to as a *good node*, there is a specific core set in the committee of such a node, and the election results agreed upon by the parties in this core set are communicable. We define a good node and discuss the communicability of election results later. In this section, we only show that when an election to a good node occurs, the lightest bins protocol is successful due to the existence of a core set. In Section 5, we analyze what could go wrong when an election is to a node that is not in this category of nodes and present our solution.

4.1 The Lightest Bins Protocol

In Figure 8, we depict the state of node A and its committee prior to the election to A from the perspective of two honest parties, p_x and p_y. p_x and p_y each have their belief of which party occupies each slot in A's committee. In the good case, A contains a core set, and if p_x and p_y are in a core set, they agree on the occupants of all of the slots in most of the nodes in A's committee. They may still disagree on the occupants of the slots in a few nodes, e.g. for those nodes whose monitoring set doesn't reliably convey their contents (as indicated for the red node).

In Algorithm 9 we describe the lightest bins protocol, which is used in the subcommittee election procedure to determine the election winners. In this procedure, p_i only communicates with and accepts messages from the parties in σ_i. Crucially, because the same party may occupy multiple slots in the committee, a party behaves as a separate entity, and is treated as a separate entity (executing a separate instance of the protocol), for each slot it occupies. Because of this, p_i includes its slot in each message it sends. When p_i receives a message from p_j containing slot s_j, p_i first verifies that $\sigma_i[s_j] = p_j$ before accepting the message.

At a high level, the lightest bins protocol starts with a predefined polylogarithmic number of bins. Each party chooses a bin uniformly at random and broadcasts their choice. The parties come to agreement on the bin choices of each of the parties. The the bin chosen by the fewest parties is the winning bin, and the parties that chose that bin are the winners.

In the LightestBins protocol, p_i participates in a single instance of BroadcastCS for each slot in the commitee. The purpose of BroadcastCS is to agree on the bin chosen by the party in the corresponding slot,

as well as which party occupies that slot. Toward this end, when a party acts as a sender in BroadcastCS (which they do if they believe they occupy the corresponding slot), they input to BroadcastCS the concatenation of their id with their bin choice. The outcome of BroadcastCS is that even if parties p_i and p_j disagree on which party occupies a given slot τ (i.e. $\sigma_i[\tau] \neq \sigma_j[\tau]$), their outputs from BroadcastCS for the instance corresponding to slot τ will be the same. This ensures that they both see the same winning bin b, since they output $*||b$ the same number of times. They also see the same list of winning parties that fill that bin. The validity property of the BroadcastCS protocol ensures that for every instance of BroadcastCS corresponding to a slot in the core set, the outcome of BroadcastCS is the input of the party occupying that slot in the core set.[10]

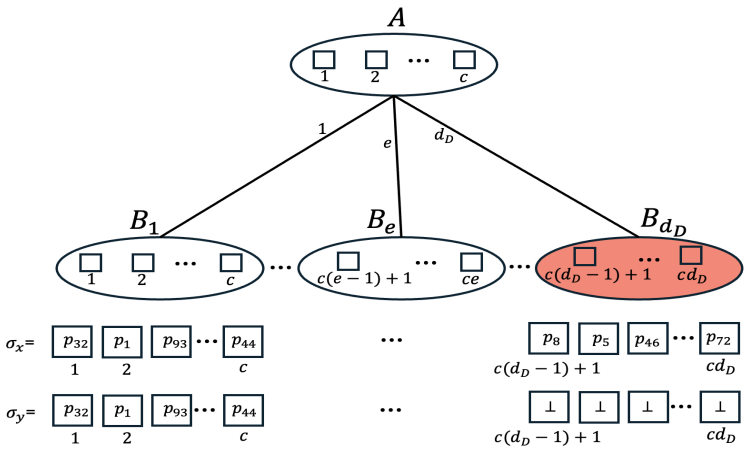

Fig. 8. The state of the committee of node A on layer $\ell > 1$ prior to the election. Each node contains exactly $c = \ln^8 n$ slots. p_x and p_y maintain arrays σ_x and σ_y of the parties they believe are in A's committee. As long as A's committee contains a core set, most of the parties in the slots are honest parties that agree on the parties occupying most of the slots. Note that A may have multiple edges to the same layer ℓ node, and for each edge there is a distinct set of numbered slots.

We would like to show that the lightest bins protocol succeeds when there is a core set. We define a core set as follows:

Definition 7 (Core Set). *Let there be a network $\mathcal{P}$ of parties, where each party p_i has an array σ_i of length $m \in \mathbb{N}^+$. The core set C is a set of tuples (p_j, in), where $p_j \in \mathcal{P}$ and $in \in [m]$ such that:*

1. $\forall (p_j, *) \in C$, p_j *is an honest party*
2. $\forall in$ *s.t.* $(*, in) \in C$, $|\{(*, in)\}| = 1$
3. $\forall (p_j, in) \in C$ *and* $\forall (p_i, *) \in C$ $\sigma_i[in] = p_j$
4. $|C| > \frac{2m}{3}$

Note that there may be multiple core sets, or no core set, for a given set of parties.

As shown in Algorithm 9, BroadcastCS is a core building block of the lightest bins protocol. In Section 4.2, we describe the BroadcastCS protocol, and show that it satisfies the properties of broadcast for a core set when a core set exists. In Section 4.3, we prove that the lightest bins protocol achieves the necessary properties under certain conditions.

[10] In the original protocol, they ensure that *exactly* c parties are elected by adding parties to the lightest bin using an arbitrary procedure. However, the honest parties are no longer choosing their bins at random, so doing so renders their calculations showing that very few honest parties are elected more than a constant number of times incorrect. We found that filling up the lightest bin is unnecessary, as long as we can show that sufficiently many honest parties are elected. An empty slot is akin to a Byzantine party, which can be tolerated as long as the total number of empty slots and Byzantine parties is less than $\frac{c}{3}s$, and the remaining slots are filled by honest parties.

Input: Each party p_i has input σ_i, an array consisting of a party's id or $\perp$ in each index s.t.

$$|\sigma_i| = \begin{cases} \frac{n \cdot \ln^8 n}{\ln^{5\ell^\star} n}, & \text{if } \ell = \ell^\star \\ \ln^{22} n, & \text{if } \ell = 0 \\ \ln^{19} n, & \text{otherwise} \end{cases} \quad (1)$$

1: let

$$num\_bins = \begin{cases} \frac{n \cdot \ln^8 n}{\ln^{5\ell^*} n}, & \text{if } \ell = \ell^\star \\ \ln^{14} n, & \text{if } \ell = 0 \\ \ln^{11} n, & \text{otherwise} \end{cases} \quad (2)$$

2: **for** each edge e from A to a layer ℓ node **do**
3: **for** each slot s in node B_e adjacent to A via edge e **do**
4: **for** every $i \in [n]$ **do**
5: **if** p_i occupies slot s **then**
6: let b be an integer chosen uniformly at random from $[num\_bins]$
7: invoke BroadcastCS as a sender with inputs $\sigma_i[s] \parallel b$, σ_i
8: **else** invoke BroadcastCS with input σ_i
9: let B be the array of outputs of BroadcastCS for each slot
10: let $b^\star$ be the bin number in B with the lowest cardinality
11: let *winners* be an array of the parties p_j ordered by id s.t. $p_j||b^*$ is in B (including multiplicities)
12: **return** $winners, b^*$

Fig. 9. LightestBins for election to node A on layer $\ell + 1$

4.2 Byzantine Broadcast for a Core Set

BroadcastCS is a protocol for Byzantine Broadcast adapted from that of [BGP92]. The protocol starts with the sender sending their input value to all of the parties. Then each party runs a Phase-King based Byzantine agreement protocol on each bit of the sender's input. We assume that each party p_i has as input an array σ_i, which corresponds to all of the slots in the nodes participating in the election. If party p_i has a non-null party p_j in position τ of σ_i, this indicates that p_i believes that p_j occupies slot τ in the list of participating nodes. We assume that c is the total number of slots in the committee participating in the election to the node.

In Lemma 7 we prove that the protocol satisfies the necessary properties of byzantine broadcast when there exists a *core set*.

In the following, we refer to lines 8-21 of the protocol as the phase-king subroutine. We refer to the king of round k as the party $\sigma[k]$. We say that a slot s is in core set C if $(*, s) \in C$.

Lemma 5. *If* BroadcastCS *is occurring for a slot s in C, and all honest parties in C begin the phase-king subroutine holding the same bit b s.t. $val = b$, then all honest parties in C end the subroutine with $val = b$.*

Proof. There are more than $2c/3$ honest parties in C , so they all receive $> 2c/3$ messages with b from parties in σ and set $c_b = 1$. As a result, all honest parties in C set $val = b$ and have $d_b > 2c/3$. Honest parties only send one message of each type, so honest parties don't change their value if the king sends a value other than b.

Lemma 6. *If* BroadcastCS *is occurring for a slot s in C, and round k of the phase-king subroutine has a king from C, then all honest parties in C have the same val by the end of that round.*

Proof. If all honest parties in C adopt the king's value, then the lemma follows from the fact that the king is honest and does not equivocate, and the fact that since the king is in C, all parties in C agree on who the king is. If some honest party in C doesn't adopt the king's value, then the king must have proposed that party's value due to the condition by which the king chooses their value and the condition by which the parties decide whether to adopt the king's value (lines 17 and 21). The lemma follows from the fact that no honest party in C could have $d_{val'} > 2c/3$ for val' other than the king's value due to quorum intersection, so all honest parties in C either adopt the king's value or already hold the king's value.

Input: the sender has γ-bit input x_s; every party p_i has input σ_i an array of length $c \in \mathbb{N}^*$, where each entry is a party's id or *null*

1: **if** $p_i = \sigma_i[s]$ **then**
2: send $\langle s_i, x_s \rangle$ to all parties in σ
3: **if** received $\langle \sigma_i[s], x \rangle$ from party $\sigma_i[s]$ **then**
4: let $v = x$
5: **else** set $v = 0$
6: **for** $\beta \in [\gamma + 1]$ **do**
7: let $val = v[\beta]$
8: **for** $k \in [\lceil \frac{c}{3} \rceil + 1]$ **do**
9: set c_1, c_0, d_1, and d_0 to 0 and send $\langle s_i, val \rangle$ to all parties in σ_i
10: **if** received $> 2c/3$ distinct messages $\langle in, b \rangle$ from parties p_j s.t. $\sigma_i[in] = p_j$ **then**
11: set $c_b = 1$ and send $\langle s_i, c_0, c_1 \rangle$ to all parties in σ_i
12: let d_0 be the count of distinct messages $\langle in, 1, * \rangle$ received from parties p_j s.t. $\sigma_i[in] = p_j$
13: let d_1 be the count of distinct messages $\langle in, *, 1 \rangle$ received from parties p_j s.t. $\sigma_i[in] = p_j$
14: **if** $d_1 > \lceil \frac{c}{3} \rceil$ **then**
15: set $val = 1$
16: **else**
17: set $val = 0$
18: **if** $s_i = k$ **then**
19: send $\langle s_i, val \rangle$ to all parties in σ_i
20: **if** received message $\langle k, val_k \rangle$ from $\sigma_i[k]$ and $d_{val} \leq 2c/3$ **then**
21: set $val = val_k$
22: set $v[\beta] = val$
23: return v

Fig. 10. BroadcastCS for slot s for party p_i occupying slot s_i

We are finally ready to prove that the protocol BroadcastCS satisfies the properties of Byzantine Broadcast for all (p, s) in C.

Lemma 7. *The* BroadcastCS *protocol satisfies the following 3 properties:*

1. *(Validity) If the broadcast is occurring for a slot s in C, then all parties in C output the input of party p, s.t. $(p, s) \in C$.*
2. *(Agreement) If honest parties p_i and p_j that are both in C output values x_i and x_j respectively, then $x_i = x_j$.*
3. *(Termination) All honest parties output a value.*

Proof. We start by proving property 1. All honest parties in C start the phase king subroutine with the sender's input. This property therefore follows from Lemma 5.

Next, we prove property 2. There must be at least one round of the phase king subroutine with an honest king in C. By Lemma 6, all honest parties in C end this round with the same value. The property therefore follows from Lemma 5.

Finally, we prove property 3. This property follows from the fact that each party sets v regardless of what they receive from the sender on line 5, and outputs v regardless of what happened in between.

4.3 LightestBins Analysis

We now prove that the lightest bins protocol achieves some desirable properties when there is a core set in the committee. Let b be the winning bin of the lightest protocol. In the following, we refer to the *winners* of the lightest bins protocol as all parties p such that the parties in the core set C [11] output $p||b$ from some instance of BroadcastCS.

First, we prove that there are sufficiently many winners from C.

[11] We define the specific core set we are interested in in Section 6.2.

Input: Each party p_i has input σ_i, an array consisting of a party's id or $\perp$ in each index indicating the parties that p_i believes occupy each slot in the committee s.t.

$$|\sigma_i| = \begin{cases} \frac{n \cdot \ln^8 n}{\ln^{5\ell^\star} n}, & \text{if } \ell = \ell^\star \\ \ln^{22} n, & \text{if } \ell = 1 \\ \ln^{19} n, & \text{otherwise} \end{cases} \quad (3)$$

1: Each party p_i initialises $W_i = \{\}$
2: Each party p_i participates in LightestBins for A_1 and has (W, aux) as the output
3: Invoke SendMsgDownTree on message on all nodes A in A_1's committee, where each party has input $\langle W, aux \rangle$
4: **if** $\ell < \ell^* - 4$ **then**
5: Invoke ConfirmElectionWinner for node A_1

Fig. 11. SubcommitteeElection to node A_1 on layer $0 < \ell < \ell^* - 1$. Note that only the parties that believe they occupy a slot in the committee of A_1 participate in this protocol.

Lemma 8. *With high probability, at least* $(1 - \frac{1}{\ln^2 n})\frac{|C| \ln^8 n}{c}$ *winners of the lightest bins protocol are from the core set* C*, where* c *is the number of slots in the committee.*

Proof. For an election to a layer 1 node, there are $\ln^{22} n$ slots in the committee, and more than $\frac{2}{3}$ of those slots contain honest parties in C. Each honest party chooses at random one of $\ln^{14} n$ bins. Using a straight forward Chernoff bound, it is clear that with high probability, the lightest bin contains at least $(1 - \frac{1}{\ln^2 n})\frac{|C| \ln^8 n}{c}$ parties from C. The case for elections on layers $\ell > 1$ follow analogously.

Lemma 9. *At most* $\ln^8 n$ *parties win the lightest bins protocol.*

Proof. This follows from the fact that a party considers a winning bin to be the one chosen by the fewest number of parties and the fact that there is a single bin choice (in other words, a single instance of BroadcastCS) per slot in the committee.

Lemma 10. *Fewer than* $\frac{1}{3}$ *of the winners are not parties from* C*.*

Proof. This follows from Lemmas 9 and 8.

4.4 Subcommittee Election

We present the full subcommittee election protocol in Figure 11. When SubcommitteeElection is invoked on each node, the parties that believe they are in the committee of that node, and only those parties, participate in the protocol. The final step of the protocol, running ConfirmElectionWinner, addresses the problems of SLE. We describe ConfirmElectionWinner in detail in Section 5.

5 Confirming Election Winners

In Section 4, we presented the subcommittee election protocol. We stated that there is a certain category of nodes, which we referred to as *good nodes*, whose subcommittee contains a specific core set. We showed that the existence of a core set implies that the lightest bins protocol is successful, and we stated that the election results as learned by the specifically defined core set of a good node are communicable.

In this section, we show how we reduce the power of the adversary to sabotage the protocol using the elections that aren't to good nodes[12]. We do so using a new procedure, ConfirmElectionWinner. Finally, we present an important lemma that upper bounds the fraction of nodes on each layer that are resistant to this procedure. We call these nodes *equivocating nodes*. Because this new bound is far lower than the bound ensured by the sampler properties of the layered network alone, we are able to ensure the completion and correctness of the overall protocol.

For the purposes of this section, we use a weak notion of bad nodes. We use a stronger definition in Section 6.2 when analyzing the entire protocol.

[12] See Section 1 for a description of how the adversary may sabotage the protocol.

Definition 8 (Weak bad node).
A node is bad if any of the following conditions are met:

1. *For a node on layer* 0*, it contains more than* $\frac{1}{3} - \varepsilon + \frac{1}{\ln^2 n}$ *fraction Byzantine parties or for a node on layer* $\ell > 0$ *it contains* $\frac{1}{3}$ *or more fraction Byzantine parties.*
2. *For node on layer* $\ell > 0$*, it contains more than* $\frac{3}{\ln n}$ *fraction bad nodes in its monitoring set.*
3. *For node on layer* $\ell > 0$*, it has more than* $\frac{1}{\ln n}$ *fraction bad nodes adjacent to it on layer* $\ell - 1$*.*

Nodes that are not weak bad nodes are good nodes. In this section, when we refer to bad nodes, we are referring to weak bad nodes.

5.1 ConfirmElectionWinner

Prior to executing ConfirmElectionWinner, each party p_i knows the bin b_i which they chose during the lightest bins protocol. They also have a belief of the winning bin, $b_i^\star$ and a belief of at most $\ln^8 n$ parties (counting multiplicities) that chose that winning bin.

The purpose of this procedure is to reduce the adversary's ability to make honest parties believe they won many elections and thereby force them to go silent. To do this, we use the monitoring sets of nodes three layers above the election node. The reason for this is that the number of nodes that are bad *and* for whom many of the monitoring sets of the nodes three layers above them contain many bad nodes is far fewer than nodes that are only bad. We call nodes that meet the latter category *equivocating nodes*, which we define more formally in Definition 9. The end result of the procedure outlined in this section is that if a node is not an equivocating node, all the adversary can do is choose a single winning bin for its election, *even if it is a bad node*. Only the honest parties that chose that bin can be confirmed to have won the election via ConfirmElectionWinner. We are then able to derive an upper bound on the number of parties on each layer that participate in even a single election to an equivocating node on the next layer. This bound is sufficiently small so that we can later use it to argue that even if all of the parties that do this go silent due to participating in shadow elections, the number of honest parties that go silent is too small to impair the liveness or correctness of the overall protocol.

We present ConfirmElectionWinner in Figure 12. We use the following notation:

- Let P be a node on layer $\ell > 0$ of the network. Then $adj^-(P)$ refers to all nodes adjacent to P on layer $\ell - 1$ (a.k.a., the nodes in P's committee)
- Let P be a node on layer $\ell < \ell^*$ of the network. Then $adj^+(P)$ refers to all nodes adjacent to P on layer $\ell + 1$ (a.k.a., the nodes for whom P is a part of their committee)

At times we abuse the notation by referring to parties $p \in adj^-(P)$ or $p \in adj^+(P)$ when we are referring to parties in the union of all nodes in $adj^-(P)$ or $adj^+(P)$. For simplicity, we use a subscript to signify application of the function multiple times, e.g. $adj_2^+(P)$ for $adj^+(adj^+(P))$. Although we use set notation, note that there is multiplicity. The same party may appear multiple times in a node, and the same node may have multiple edges to another node.

At the start of ConfirmElectionWinner, each party p_i in P_1's subcommittee initializes a variable $confirmed_i$ to 0. Later, as it receives confirmations that it won the election, it increments this variable. It considers its status as a winner *confirmed* upon receiving a threshold number of confirmations. Also at the start of the protocol, every party $p_k \in ms(adj_3^+(P_1))$ initializes a variable $winners_k$ to $\emptyset$ and $bin = \perp$. The former variable will refer to the list of parties that p_k learns are the winners of the election to P_1, and the latter refers to the bin that p_k learns was the winning bin. Following this, every party p_i in P_1's subcommittee that has $b_i \neq b_i^\star$ (p_i does not believe it won the initial election) returns false from the procedure, ensuring that only parties that believe they won the initial election may be confirmed as winners. The remaining parties in P_1's subcommittee each choose a set $\mathcal{P}(P_4)$ uniformly at random of $\ln^8 n$ nodes in $ms(P_4) \forall P_4 \in adj_3^+(P_1)$, and send a message $\langle$"poll", $P_1\rangle$ to all parties in the nodes $\mathcal{P}(P_4)$. This step introduces a polling mechanism for election winner confirmation, maintaining the desired communication complexity of the overall protocol.

Next, the parties in $ms(adj_3^+(P_1))$ learn the winners of the election to P_1. The parties in a designated node in $ms(P_1)$ tell the parties in a designated set of nodes in $ms(adj_3^+(P_1))$ the winners of the election. A

Input: $\forall$ parties p_i in nodes $A \in adj^-(P_1)$ let b_i be the bin that p_i chose and let $b_i^\star$ be the bin that p_i believes won the initial election

All parties p_i in nodes $A \in adj^-(P_1)$ initialize $confirmed_i = 0$
All parties $p_k \in ms(adj_3^+(P_1))$ initialize $winners_k = \emptyset$ and $bin = \perp$
for $p_i \in adj^-(P_1)$ **do**
  if $b_i \neq b_i^\star$ **then**
    return $false$
  else
    for $P_4 \in adj_3^+(P_1)$ **do**
      Select a set $\mathcal{P}(P_4)$ of $\ln^8 n$ nodes uniformly at random from $ms(P_4)$
      Send $\langle$"poll", $P_1\rangle$ to all parties in $\mathcal{P}(P_4)$
for $P_4 \in adj_3^+(P_1)$ **do**
  Every party p_j in the k-th node of $ms(P_1)$ tells every party in nodes $k \cdot \ln^{15} n \ldots (k+1) \cdot \ln^{15} n - 1$ of $ms(P_4)$[a] a tuple (W_j, b_j), where W_j is the list of winners and b_j is the winning bin for the election to P_1
  for $p_k \in ms(P_4)$ **do**
    if heard the same tuple (W, b) of $\leq \ln^8 n$ initial election winners W and winning bin b from $> \frac{2}{3}$ of the parties in the corresponding node in $ms(P_1)$ **then**
      p_k sets $winners_k$ equal to W, $bin = b$
      p_k sends a message ("confirmed", bin) to all parties in $winners_k$ from which it received a message $\langle$"poll", $P_1\rangle$
for $p_i \in adj^-(P_1)$ **do**
  for each node $P_3 \in adj_2^+(P_1)$ **do**
    if p_i received $\langle$"confirmed", $b_i\rangle$ from $> \frac{2}{3}$ of the parties in $\mathcal{P}(P_4)$ for $> \frac{2}{3}$ of nodes $P_4 \in adj^+(P_3)$ **then**
      p_i increments $confirmed_i$
  if $confirmed_i > \frac{2\ln^{12} n}{3}$ **then** ▷ because $|adj_2^+(P_1)| = \ln^{12} n$
    return $true$
  else return $false$

[a] (note that $|ms(P_4)| = \ln^{15} n \cdot |ms(P_1)|$)

Fig. 12. ConfirmElectionWinner for election to node P_1 on layer $\ell < \ell^* - 4$

party $p_k \in ms(adj_3^+(P_1))$ then sets $winners_k$ to be the valid list of parties (in this case, one consisting of at most $\ln^8 n$ parties) that it received from more than $\frac{2}{3}$ of the parties in the corresponding node in $ms(P_1)$ and bin to be the bin number it received from more than $\frac{2}{3}$ of the parties in the corresponding node in $ms(P_1)$. Upon setting $winners_k$, p_k then sends a message $\langle$"confirmed", $bin\rangle$ to all of the parties in $winners_k$ from which it received a message $\langle$"poll", $P_1\rangle$.

Each party p_i in P_1's committee considers that it has received confirmation as follows. p_i counts the messages it received from parties in $ms(adj_3^+(P_1))$ that it has chosen to poll. Specifically, it considers the count of messages $\langle$"confirmed", $b_i\rangle$ it received from parties in $\mathcal{P}(P_4)$ for the nodes $P_4 \in adj^+(P_3)$ for each node P_3 in $adj_2^+(P_1)$ individually. It increments $confirmed_i$ once for a node P_3 if it received $\langle$"confirmed", $b_i\rangle$ from more than $\frac{2}{3}$ of the parties in $\mathcal{P}(P_4)$ more than $\frac{2}{3}$ of the nodes $P_4 \in adj^+(P_3)$. If p_i increments $confirmed_i$ sufficiently many times, it returns $true$ from the procedure, considering its status as an election winner as *confirmed*. Otherwise, p_i considers itself as not having been confirmed as an election winner, returning $false$.[13]

ConfirmElectionWinner Analysis We proceed to prove that the election confirmation procedure achieves the aforementioned result with the desired communication complexity. Due to the presence of shadow elections, we specify that for an election to a good non-equivocating node there is a specific list of correct winning parties and winning bin that we are interested in. We use the following assumption in our proofs.

[13] For simplicity, we do not refer to slots in this section, but note that parties must receive confirmation separately for each slot that they occupy in the subcommittee.

Assumption 1 *We refer to W_C and $b^\star$ as the correct initial election winners for P_1. If P_1 is a non-equivocating good node, then all but $1/\ln n$ fraction of the nodes in $ms(P_1)$ know the correct initial election winners for the election to P_1, i.e. $W_j = W_C$ and $b_j = b^\star$. The list W_C includes a list of parties p_i s.t. $b_i = b^\star$.*[14]

When analyzing the full protocol in Section 6.2, we specify what we mean by the correct election winners (Remark 1) and show that this assumption holds.

For the purposes of the following proofs, we define a node P_1 as a **non-equivocating node** as follows.

Let P be a node in the network. $X_4(P) = true$ if at most $\frac{3}{\ln n}$ fraction of the nodes in $ms(P)$ are bad nodes. $X_3(P) = true$ if $|\{P_4 | P_4 \in adj^+(P) \text{ and } X_4(P_4) = true\}| > \frac{2}{3}|adj^+(P)|$. $X_2(P) = true$ if $|\{P_3 | P_3 \in adj^+(P) \text{ and } X_3(P_3) = true\}| > \frac{2}{3}|adj^+(P)|$.

Definition 9 (Non-equivocating node). *Let P_1 be a node on layer $\ell > 0$ of the network. If $|\{P_2 | P_2 \in adj^+(P_1) \text{ and } X_2(P_2) = true\}| > \frac{2}{3}|adj^+(P_1)|$, then P_1 is a non-equivocating node.*

If a node on layer $\ell > 0$ is *not* a non-equivocating node, we refer to it as an **equivocating node**.

We start by proving that for each non-equivocating node P_1, the adversary is forced to choose a single bin s.t. the parties that chose that bin believe they won the election, even if P_1 is a bad node. Moreover, other parties that did not choose this bin will know that they did not win the election.

Lemma 11. *Let P_1 be a non-equivocating node. If honest parties p_i and p_j chose bins b_i and b_j and they both return true from* Confirm-Election-Winner *for the election to P_1, then $b_i = b_j$.*

Proof. In order for a party $p_i \in adj^-(P_1)$ to return *true* from Confirm-Election-Winner for the election to P_1, it must have $confirmed > 2\ln^{12} n/3$ (line 20). p_i increments $confirmed$ at most once for each node $P_3 \in adj^+(adj^+(P_1))$ if it receives a message ("confirmed", b_i) from more than 2/3 of the parties in more than 2/3 of the nodes in $\mathcal{P}(P_4)$ for more than 2/3 of the nodes $P_4 \in adj^+(P_3)$ (lines 18- 19).

Consider a single node P_4 s.t. at most $\frac{3}{\ln n}$ fraction of the nodes in $ms(P_4)$ are bad. We show that there can be at most a single unique bin b such that parties p_i with $b_i = b$ receive *preliminary confirmation* from $ms(P_4)$. We say that p_i receives preliminary confirmation from P_4 if it receives ("confirmed", b_i) from $> 2/3$ of the parties in $> 2/3$ of the nodes in $\mathcal{P}(P_4)$. Each honest party only sends messages of the type ("confirmed", b) for a single, unique bin b. Assume that a party p_i has received preliminary confirmation from $ms(P_4)$. Then, of the $\ln^8 n$ nodes in $\mathcal{P}(P_4)$, from more than $\frac{2\ln^8 n}{3}$ of those nodes had more than 2/3 of the parties send messages of the type $\langle$"confirmed", b_i $\rangle$ to p_i. At least $1 - \frac{3}{\ln n}$ fraction of those nodes must have been good nodes, so at least $\frac{2}{3} - \frac{3}{\ln n}$ fraction of the nodes in $\mathcal{P}(P_4)$ are good nodes in which $> 2/3$ of the parties sent $\langle$"confirmed", b_i $\rangle$ to p_i. Since p_i chose the nodes in $\mathcal{P}(P_4)$ uniformly at random, it follows from a Chernoff bound that with high probability, at least $\frac{2}{3} - \frac{4}{\ln n} > \frac{1}{2}$ fraction of the nodes in $ms(P_4)$ must be good nodes in which more than $\frac{1}{3}$ fraction of the parties are honest parties with $bin = b_i$. It follows from a Chernoff bound and a quorum intersection argument on the honest parties in a good node in $ms(P_4)$ that another honest party p_j with $b_j \neq b_i$ cannot receive preliminary confirmation from P_4.

Next, consider a single node P_3 such that more than $\frac{2}{3}$ of the nodes $P_4 \in adj^+(P_3)$ have at most $\frac{3}{\ln n}$ fraction bad nodes in $ms(P_4)$. We show that there can be at most a single unique bin b such that parties p_i with $b_i = b$ receive *secondary confirmation* from P_3. We say that p_i receives secondary confirmation from P_3 if it receives preliminary confirmation from the monitoring sets $ms(P_4)$ of more than $\frac{2}{3}$ of the nodes $P_4 \in adj^+(P_3)$. By the definition of P_3, more than 2/3 of the nodes $P_4 \in adj^+(P_3)$ have the property that preliminary confirmation can only be received from $ms(P_4)$ for a single bin (as shown in the previous paragraph). From quorum intersection on the nodes $P_4 \in adj^+(P_3)$, it follows that for such a node P_3, secondary confirmation can be received from P_3 for a single bin. This is because for more than $\frac{2}{3}$ of the nodes $P_4 \in adj^+(P_3)$, $ms(P_4)$ will only give preliminary confirmation for a single bin.

Finally, because P_1 is a non-equivocating node, by the definition of a non-equivocating node, it follows that more than $\frac{2}{3}$ of the nodes $P_3 \in adj_2^+(P_1)$ give secondary confirmation for only a single bin b. If a party p_i

[14] Note that it's possible for a party to win a *shadow election* to a non-equivocating good node if they learned the parties in the committee from a monitoring set which conveyed the wrong information. In this case, we don't consider this party as a correct initial election winner, regardless of its bin choice, because it didn't participate in the "actual election" to the node. We specify what we consider the correct initial election winners in Section 6.2, in Remark 1.

increments $confirmed_i$ for a node P_3, then they have received secondary confirmation from P_3 (lines 18-19). The lemma therefore follows from quorum intersection on the nodes $P_3 \in adj_2^+(P_1)$ from which p_i and p_j received secondary confirmation, causing them to increment $confirmed_i$ and $confirmed_j$ respectively.

Next we prove a lemma showing that for all good nodes that are also non-equivocating, the election winner confirmation procedure does not change the initial election results. Parties that won the initial election, and only parties that won the initial election, are all confirmed as winners.

Lemma 12. *If P_1 is a non-equivocating* good *node, a party p_i returns true from* ConfirmElectionWinner *for the election to P_1 iff $b_i = b_i^\star$ and p_i is in W_C with high probability.*

Proof. Observe that by the condition on line 5, we trivially have that if $b_i \neq b_i^\star$ a party returns $false$ from ConfirmElectionWinner. All that remains to prove is that all parties p_i for whom $b_i = b_i^\star$ return $true$ from ConfirmElectionWinner.

First, we show that for all nodes $P_4 \in adj_3^+(P_1)$, the parties p_k in at most $4/\ln n$ fraction of the nodes in $ms(P_4)$ do not learn the correct initial election winners from the parties in the corresponding node in $ms(P_1)$, causing them to set $winners_k$ and bin incorrectly (or never set them to anything) on lines 13-15. To see this, for parties in a node in $ms(P_4)$ to learn incorrect or no information, the corresponding node in $ms(P_1)$ must be bad or the honest parties in the node must have learned an incorrect list of initial election winners by the conditions on lines 13-15). Since P_1 is a good node, at most $3/\ln n$ fraction of the nodes in $ms(P_1)$ can be bad nodes, and at most $1/\ln n$ fraction of the nodes in $ms(P_1)$ could have learned incorrect initial election winners (by Assumption 1). It follows that no more than $4/\ln n$ fraction of the nodes in $ms(P_4)$ for all nodes $P_4 \in adj_3^+(P_1)$ can have their corresponding nodes in $ms(P_1)$ be bad or have learned the incorrect initial election winners, thus proving the statement.

By the definition of a non-equivocating election node, more than $\frac{2}{3}$ of the nodes $P_2 \in adj^+(P_1)$ have the property that more than $\frac{2}{3}$ fraction of the nodes $P_3 \in adj^+(P_2)$ with more than $\frac{2}{3}$ of the nodes $P_4 \in adj^+(P_3)$ having at most $\frac{3}{\ln n}$ fraction bad nodes in $ms(P_4)$. By the statement proved in the preceding paragraph, all such nodes P_4 have the property that more than $1 - \frac{3}{\ln n} - \frac{4}{\ln n}$ fraction of the nodes in $ms(P_4)$ are good nodes in which the honest parties learned the correct initial election winners. It follows from a Chernoff Bound that with high probability, for every good node P_4 that more than 2/3 of the nodes in $\mathcal{P}(P_4)$ are good nodes in which the parties know the correct initial election winners, so all honest parties in W_C increment $confirmed$ more than $\frac{2\ln^{12} n}{3}$ times, thus proving the lemma.

The following lemma bounds the fraction of equivocating nodes per layer.

Lemma 13. *For every layer ℓ s.t. $0 < \ell < \ell^* - 4$, at most $\frac{2^{28}\cdot 5^8}{\ln^{20} n}$ fraction of the nodes on layer ℓ are equivocating election nodes.*

Proof. We start the proof by showing a bound on the number of nodes A on any given layer $\ell' = \ell - 3$ such that at least $\frac{1}{3}$ fraction of the nodes in $adj^+(A)$ have more than $3/\ln n$ fraction of bad nodes in their monitoring sets.

Let M be the set of nodes on layer ℓ' and P be the set of nodes on layer $\ell' + 1$. By the construction of the layered network, $|M| = n/ln^{5\ell'+5}n$ and $|P| = n/ln^{5\ell'+10}n$. Additionally, the degree from nodes in M to nodes in P is $d_{up} = \ln^6 n$, and the degree from nodes in P to nodes in M is $d_{down} = \ln^{11} n$. We know from Lemma 14 below that at most $\frac{400}{\ln^2 n}$ fraction of the nodes in P are bad nodes due to having more than $3/\ln n$ fraction of bad nodes in their monitoring set (property 2 of Definition 8). Let S be the set of bad nodes that are bad due to satisfying this property in P and let $T \subseteq M$ be the set of nodes on layer ℓ' s.t. each node from T has at least 1/3 of its neighbors in S. By definition of the set T, $e(S,T) \geq |T|d_{up}/3$, where $e(S,T)$ is the number of edges between S and T.

Using property 5 of the layered network from Lemma 1, we have

$$
\begin{aligned}
e(S,T) &\le \frac{d_{up}|S||T|}{|P|} + \lambda\sqrt{|S||T|} + \max\{|S|,|T|\} \\
\frac{|T|d_{up}}{3} &\le \frac{400d_{up}|T|}{\ln^2 n} + 10\ln^{5.5} n\sqrt{|S||T|} + \max\{|S|,|T|\} && \text{(Using } |S|/|P| \le 400/\ln^2 n.\text{)} \\
\frac{d_{up}}{3} &\le \frac{400d_{up}}{\ln^2 n} + 10\ln^{5.5} n\sqrt{\frac{|S|}{|T|}} + \max\left\{\frac{|S|}{|T|},1\right\} \\
\frac{\ln^6 n}{4} &\le 10\ln^{5.5} n\sqrt{\frac{|S|}{|T|}} + \max\left\{\frac{|S|}{|T|},1\right\} && \text{(for large enough } n\text{)} \\
\sqrt{|T|} &\le \frac{40}{\ln^{0.5} n}\sqrt{|S|} + \frac{4\sqrt{|T|}}{\ln^6 n}\max\left\{\frac{|S|}{|T|},1\right\}
\end{aligned}
$$

If $\max\left\{\frac{|S|}{|T|},1\right\} = 1$, then

$$\sqrt{|T|} \le \frac{40}{\ln^{0.5} n}\sqrt{|S|} + \frac{4\sqrt{|T|}}{\ln^6 n} \implies \frac{1}{2}\sqrt{|T|} \le \frac{40}{\ln^{0.5} n}\sqrt{|S|} \implies |T| \le \frac{6400}{\ln n}|S|$$

else,

$$
\begin{aligned}
\sqrt{|T|} &\le \frac{40}{\ln^{0.5} n}\sqrt{|S|} + \frac{4\sqrt{|T|}}{\ln^6 n}\frac{|S|}{|T|} \\
\sqrt{\frac{|T|}{|S|}} &\le \frac{40}{\ln^{0.5} n} + \frac{4}{\ln^6 n}\sqrt{\frac{|S|}{|T|}} \\
\frac{1}{2}\sqrt{\frac{|T|}{|S|}} &\le \frac{40}{\ln^{0.5} n} && \text{(for large enough } n\text{)} \\
|T| &\le \frac{6400}{\ln n}|S|
\end{aligned}
$$

Thus, in both the cases, we have $|T| \le \frac{6400}{\ln n}|S|$ which implies

$$|T| \le \frac{6400 \cdot 400|M|}{\ln^8 n} = \frac{16 \cdot 400^2|M|}{\ln^8 n}.$$

We apply the same line of reasoning twice more to derive the bound on the fraction of equivocating nodes on a given layer in the lemma statement.

First, we bound the number of nodes from layer $\ell' + 2$ with at least $1/3$ of its neighbors in T. Call this set of nodes T_2. This will give us an upper bound on the number of nodes A on layer $\ell' + 2$ with more than $\frac{1}{3}$ fraction nodes $A' \in adj^+(A)$ having more than $\frac{1}{3}$ fraction of the nodes $A'' \in adj^+(A')$ with more than $\frac{3}{\ln n}$ fraction bad nodes in their monitoring sets. Applying the same expander argument as above but now replacing the role of S and P with T and M respectively, we get that $\frac{|T_2|}{|R_{\ell'+2}|} \le 16^2 \cdot 400^3/\ln^{14} n$ (note that $|R_\ell|$ is the number of nodes in layer ℓ). Finally, making the same argument again, now with the sets T_2 and $R_{\ell'+2}$, we get the required bound of $16^3 \cdot 400^4/\ln^{20} n$ on the fraction of equivocating nodes from layer $\ell' + 3 = \ell$ as claimed in the lemma statement.

Lemma 14. *At most $\frac{400}{\ln^2 n}$ fraction of the nodes on layer ℓ for $1 < \ell < \ell^\star$ have more than $\frac{3}{\ln n}$ fraction bad nodes in their monitoring set.*

Proof. By property 3 of Lemma 1, at most $400/\ln^2 n$ fraction of the layer 0 nodes are bad due to satisfying property 1 of Definition 8. That is, up to $\frac{400n}{\ln^7 n}$ layer 0 nodes are bad. The communication tree rooted at a node on layer ℓ has $\ln^{5\ell} n$ leaf nodes. It follows that at most $\lfloor \frac{400n}{\ln^7 n} \div \ln^{5\ell-1} n \rfloor \le \frac{n}{\ln^{5\ell+6} n}$ nodes on layer

ℓ may have more than $\frac{1}{\ln n}$ fraction bad leaf nodes for their communication trees. Because layer ℓ contains $\frac{n}{\ln^{5\ell+5} n}$ nodes, it follows that at most $\frac{1}{\ln n}$ fraction of the nodes on layer ℓ may have more than $\frac{1}{\ln n}$ fraction bad leaves in their communication trees. Refer to this set of layer ℓ nodes as the set T.

By property 3 of Lemma 1, it follows that at most $\frac{400}{\ln^2 n}$ fraction of the nodes on layer $\ell+1$ are adjacent to more than $\frac{1}{\ln n} + \frac{1}{\ln^2 n}$ fraction nodes from T. It follows that at most $\frac{400}{\ln^2 n}$ fraction of the nodes on layer $\ell+1$ have more than $\frac{1}{\ln n} + \frac{1}{\ln n} + \frac{1}{\ln^2 n} < \frac{3}{\ln n}$ fraction bad nodes in their monitoring set.

Finally, we show that the election confirmation procedure does not add more than polylogarithmic bits of communication per process.

Lemma 15. Confirm-Election-Winner *incurs polylogarithmic per process communication complexity.*

Proof. Consider the election to a single node P_1 on layer ℓ. Communication begins on line 11. Each party in the ith node of $ms(P_1)$ sends a list of $O(\ln^8 n)$ parties to every party in $\ln^{15} n$ nodes in $ms(P_4)$ for all nodes $P_4 \in adj_3^+(P_1)$. Since there are $\ln^{11} n$ parties per layer 0 node, a party in a node in $ms(P_1)$ sends a list of election winners to at most $\ln^{11} n \cdot \ln^{15} n \cdot |adj_3^+(P_1)|$. Note that $|adj_3^+(P_1)| = \ln^{18} n$. Every party is in at most $\ln^6 n$ layer 0 nodes, so each party sends a list of polylogarithmically many parties to polylogarithmically many parties for a given election.

Next, we consider the communication on line 15. Every party in $ms(P_4)$ sends a message "confirmed" to at most $\ln^8 n$ parties. $|adj_3^-(P)| \leq \ln^{33} n$ for all nodes P, so parties in $ms(P_4)$ perform election confirmation for polylogarithmically many nodes on layer ℓ. The remainder of the lemma follows from the fact that every party is in $\ln^6 n$ layer 0 nodes and the fact that parties poll only $\ln^8 n$ nodes in each monitoring set $ms(P_4)$.

6 Leader Election with Sublinear Per Party Communication Complexity

In this section, we present our analysis of the full protocol with an updated proof of its correctness. In Section 6.1, we present lemmas that are necessary to show that the protocol completes correctly with sublinear per party communication complexity. Recall that the protocol uses a silencing mechanism to ensure that parties don't get elected to too many nodes per layer, blowing up the communication complexity. However, this silencing mechanism could also harm the completion of the protocol if too many honest parties go silent. Toward this end, we show a bound, for each layer of the network, on the fraction of nodes containing many silent honest parties.

Finally, in Section 6.2, we analyse the full protocol. We show that with high probability, the protocol elects a supreme committee of polylogarithmic size containing more than 2/3 fraction honest parties, and that all but $o(1)$ of the network is aware of. In addition, we show that the protocol satisfies the necessary communication complexity and round complexity.

6.1 Bounding the Negative Effects of Silenced Parties

In this section, we show that for most nodes on each layer of the network, very few parties in the node are honest parties that have gone silent (Lemma 17). A crucial part of this proof is our bound on the number of equivocating nodes per layer proved in the preceding section in Lemma 13. We start by introducing the notion of a *potentially good node*. We introduce this notion because we wish to bound the fraction of silent honest parties in the nodes on each layer. However, we haven't yet argued that the elections complete correctly on each layer; we do this in Section 6.2. In this section, we only bound the fraction of nodes per layer that may be *potentially good* but contain many silent honest parties. In Section 6.2 we show that most nodes on each layer are in fact *potentially good nodes*, and we use the bounds proved in this section to analyse the full protocol.

Definition 10 (Potentially good node). *A potentially good node is a non-equivocating node whose committee contains a core set.*

As a consequence of Lemma 8, potentially good nodes contain more than $\frac{2\ln^8 n}{3} > \ln^7 n$ (for sufficiently large n) parties with high probability. We also use the notion of *latently silent honest parties* in our proofs, which we motivate and define next.

In order to prove the desired bound on the number of nodes per layer containing many silent honest parties, we need to bound the number of silent honest parties on each layer, counting multiplicities. Each layer ℓ has a certain fraction of equivocating nodes. Since there is no guarantee on the outcome of the election to an equivocating node, every party that participates in an election to an equivocating node may believe that they were elected. Since they may participate in *shadow elections* from that node onwards (which the adversary can ensure that they win), they may go silent for all non-shadow elections that they win on the next layer. Rather than immediately choosing to silence such honest parties on the next layer, the adversary could "collect" many honest parties by having them believe they are elected to equivocating nodes, making them *latently silent honest parties*, and use them to sabotage the protocol on some higher layer.

Definition 11 (Layer ℓ latently silent honest party). *A party is a latently silent honest party in node A for layer ℓ if it is (elected to) in an equivocating node on layer ℓ' such that $\ell' < \ell$ and it knows that it is elected to node A.*

On the one hand, the adversary may choose to silence a latently silent honest party by having it win all the elections to (equivocating and non-equivocating) nodes adjacent to an equivocating node that it is in (via shadow elections). In this way, the adversary may immediately silence this party in all other layer $\ell + 1$ nodes to which it wins elections (since we assume the adversary can choose the winning bin for all elections). But if a party is silent in a node, it will not participate in elections to the next layer connected to that node. To be thorough in deriving our upper bound on the number of nodes containing many silent honest parties on each layer, we must account for the fact that the adversary may *not* have this party go silent in all other layer $\ell + 1$ nodes to which it wins elections. The adversary can choose to have this party win fewer than 10 shadow elections and elections to equivocating nodes, so that it is *not* silent for all of the other nodes to which it wins elections. This party can carry on up the layered network, winning a few shadow elections and elections to equivocating nodes on each layer and a few elections to non-equivocating nodes on each layer. Basically, we consider that once an honest party participates in an election to an equivocating node, the adversary can silence this party in all nodes to which it wins an election at any point up the tree. We call these parties "latently silent" because the adversary may choose, at some point higher in the network, to silence these parties by having them win 10 shadow elections. At first glance, it seems like the adversary could turn many honest parties into latently silent honest parties and then silence a huge fraction of the parties on some higher layer of the network, causing the protocol not to terminate. However, in Lemma 16, we show that most nodes on each layer contain a very small fraction of latently silent honest parties. This ensures that even if the adversary silences all latently silent honest parties at once on any given layer of the network, it still holds that most nodes per layer don't contain too many silent honest parties.

The distinction in the definition that a party must know it is elected to node A is because the harm done by a latently silent honest party is that it may be elected up the tree, and go silent at some point later that is chosen by the adversary. If the party doesn't know it is elected to node A, it will not participate in elections to nodes adjacent to A on the next layer. [15] We assume that all parties in the committee of an equivocating node believe they are elected to that equivocating node. In the proof of Lemma 16, we consider latently silent honest parties as only coming from potentially good nodes and equivocating nodes. This is because when analysing the whole protocol in Section 6.2, we implicitly assume that all of the parties coming from the remaining nodes (those that are neither equivocating nor potentially good) are full of Byzantine parties, and Byzantine behavior subsumes the negative effects of latently silent honest parties.

Lemma 16 shows a bound on the number of nodes on each layer of the network that contain many *latently silent honest parties*. In Lemma 17, we bound the fraction of silent honest parties for most nodes per layer. The proofs of the following two lemmas may be found in the full version of the paper.

Lemma 16. *At most $\frac{400}{\ln^2 n}$ fraction of the layer ℓ nodes are potentially good nodes with more than $\frac{404\ell}{\ln^2 n}$ fraction layer ℓ latently silent honest parties for $1 < \ell < \ell^*$ with high probability.*

Lemma 17. *At most $\frac{412}{\ln^2 n}$ fraction of the nodes on layer ℓ are potentially good nodes in which more than $\frac{404\ell+1}{\ln^2 n}$ fraction of the parties are honest parties that have gone silent for $0 < \ell < \ell^* - 4$, with high probability.*

[15] This can happen, for example, if a party is not in a core set (Definition 7) in the committee of the node to which the election is occurring. This implies that they may not learn the results of the election, and therefore that they are elected.

6.2 Full Protocol Analysis

In this section, we analyze the correctness and communication complexity of the entire protocol. We start by defining some useful terms. In our proofs, we use the notions of good and bad nodes and good and bad leaves which we define below.

Definition 12 (Bad node). *If node A is a layer 0 node, then A is a bad node if it contains fewer than $(\frac{2}{3} + \varepsilon - \frac{1}{\ln^2 n})\ln^{11} n$ honest parties. If node A is a layer 1 node, then A is a bad node if it contains fewer than than $f_1 \ln^8 n$ honest parties, where $f_1 xxZSDsdszddsss = \frac{2}{3} + \varepsilon - \frac{403}{\ln^2 n}$.*

If A is a node on layer $\ell > 1$, let $B_1 \dots B_d$ be the nodes in A's commitee (including multiplicities). A is a bad node if at least one of the following conditions holds:

1. *More than $\frac{814}{\ln^2 n}$ fraction of the nodes $B_1 \dots B_d$ are bad.*
2. *The fraction of bad leaves from the trees $B_1 \dots B_d$ is more than $\frac{813\ell+400}{\ln^2 n}$.*
3. *A is an equivocating node.*
4. *A contains at least one of the following:*
   (a) *$\frac{1}{3}$ or more fraction parties that are Byzantine.*
   (b) *Fewer than $(f_{\ell-1} - \frac{404(\ell-1)+816}{\ln^2 n})\ln^8 n$ honest parties.*
   (c) *More than $\frac{404\ell+1}{\ln^2 n}$ fraction silent honest parties.*

If a node is not a bad layer ℓ node, it is a good layer ℓ node. Note that nodes that are weak bad nodes (Definition 8) are also bad by this stronger definition. Due to shadow elections, some parties may have different views of which parties won the election to a node, and therefore make up that node's subcommittee. We clarify via Definition 14 and Remark 1 below which parties we consider to be the election winners of a node.

Recall that the leaves of the communication tree are the layer 0 nodes. Let A be a layer 0 node that is a descendant of layer ℓ node B in the communication tree. Then $path_{A,\ell}$ refers to the list of nodes along the path from layer 0 node A to layer ℓ node B (inclusive of both A and B). We define good and bad leaves as follows.

Definition 13 (Bad leaf for layer ℓ). *If a layer 0 node is bad, it is a bad leaf for layer 0. A layer 0 node A is a bad leaf for layer ℓ if there is a bad node in $path_{A,\ell}$.*

If a layer 0 node is not a bad leaf for layer ℓ, it is a good leaf for layer ℓ. If node B_ℓ is the layer ℓ node for which node B_0 is a layer 0 descendant in the communication tree, then we also say that B_0 is a bad/good leaf for B_ℓ.

Proof Outline. Our ultimate goal is to show that there is a supreme committee of parties elected that contains more than $\frac{2}{3}$ honest parties and that all but $o(1)$ of the parties in the network know which parties are in the supreme committee. Since the supreme committee is elected among the parties in the nodes on the penultimate layer, this requires that all but very few nodes on the penultimate layer are good nodes. For this reason, we go layer by layer up the network, and prove that most nodes on each layer are good nodes. Intuitively, a good node contains sufficiently many non-silent honest parties and less than $\frac{1}{3}$ fraction Byzantine parties, satisfying the necessary threshold for conducting elections. For a node to be a good node, the election to that node must occur successfully, electing more than $\frac{2}{3}\ln^8 n$ honest parties. This requires that the node has many good nodes in its committee and that the non-silent parties in the good nodes in its committee form a core set[16]. In Lemmas 18 and 19, we show that this is the case for a node with sufficiently many good nodes in its committee. In Lemma 20, we show that if these properties are satisfied for a node, sufficiently many honest parties are elected to the node. This leaves one last piece to show that most nodes per layer are good nodes: a good node must not have too many *silent honest parties*. Upon showing that the elections to most nodes per layer occur successfully, we use the calculations from Section 6.1 which bound the fraction of nodes per layer that contain many silent honest parties. With this, we are able to prove that most nodes on a given layer are good nodes. This in turn enables us to bound the fraction of bad leaves

[16] See Section 4 for the election protocol and core sets.

per layer. The significance of good and bad leaves is that the leaf nodes make up the monitoring sets, which are used to convey the contents of a node's subcommittee to the other nodes. For this to occur correctly, a node should have many good leaves. In Lemma 21, we show that the necessary bound on the fraction of bad nodes and bad leaves per layer holds, and we ultimately show that the supreme committee is elected successfully in Lemma 22. Finally, in Lemmas 23 and 24, we show that the full protocol satisfies the necessary communication complexity and round complexity.

We now proceed with the full protocol analysis. Let A be a layer ℓ node for $\ell > 0$ such that at most $\frac{814}{\ln^2 n}$ fraction of the nodes adjacent to A on layer $\ell - 1$ are bad nodes. In Lemma 19 we show that these parties form a core set. Furthermore, the winners of the election run among the honest parties in these nodes form the subcommittee of node A. Addressing Assumption 1 in Section 5, we consider the correct election winners for the election to a good node A as follows.

Definition 14 (Communicable core set of node A, $cs_G(A)$). *We refer to the* communicable core set *of A, $cs_G(A)$, as the honest parties in the good nodes in A's committee that are not silent.*

Remark 1. The correct election winners of node A are the winners of the election run among the parties in $cs_G(A)$.

If the non-silent honest parties in the good nodes in the committee of A do not form a core set, the correct election winners of the node is undefined. This is okay because if a node does not contain such a core set, it is a bad node by definition and we don't make any guarantees regarding the outcome of its election. In the proof of Lemma 21, we show that Assumption 1 holds. [17]

We start with a lemma stating that if a node is good, then honest parties in the monitoring set of A know the parties in A's subcommittee.

Lemma 18. *Let A be a good node on layer ℓ s.t. $0 \leq \ell \leq \ell^*$. Then every honest party in every node in $ms(A)$ that is a good leaf node for layer $\ell - 1$ knows the parties in A's subcommittee.*

Proof. By the definition of a good leaf node, every node along the path from a layer $\ell - 1$ node to a good leaf node for layer $\ell - 1$ is good. By the definition of a good node, each of these nodes contains more than $\frac{2\ln^8 n}{3}$ honest parties. Observe also that every node on the path to a good leaf node contains a good node, and thus a child node knows the exact list of parties in the parent node. It follows that every honest party in every node along the path to a good leaf node hears the correct list of winning parties from a majority of the parties in the node above it on the path during SendMsgDownTree, thus proving the lemma.

Lemma 19. *Let A be a layer ℓ node, where $0 < \ell \leq \ell^*$, such that at most $\frac{814}{\ln^2 n}$ fraction of the nodes in A's committee are bad. Then the non-silent honest parties in the good nodes in A's committee form a core set with high probability.*

Proof. Consider a good node B on layer $\ell - 1$. There are at least $(f_{\ell-2} - \frac{404(\ell-2)+816-404\ell}{\ln^2 n})\ln^8 n$ non-silent honest parties in B. For some $\varepsilon = O(\frac{1}{\ln \ln n})$, this is more than $\frac{2\ln^8 n}{3}$. In addition, every honest party in all but $\frac{813\ell+400}{\ln^2 n}$ fraction of the nodes in $ms(B)$ are in good leaves for their respective layer-$\ell - 2$ trees and know the correct list of parties in B's subcommittee by Lemma 18. Next, we argue that LearnProcessors run between any two good nodes B and B' in A's committee is successful. That is, the parties in B learn the correct parties in B' and vice versa. Consider the ith node in $ms(B)$, M, and the corresponding ith node in $ms(B')$, M'. In order for the parties in M to have the wrong list of parties in B', either B or B' must be bad leaves. So with probability $(1 - \frac{813\ell+400}{\ln^2 n})^2$, both M and M' are good leaves for B and B', respectively.

[17] We take a moment to address the phenomenon of *oblivious* honest parties. Recall from Section 4 that only the parties in the core set of the committee participating in an election are guaranteed to learn the correct outcome of that election. Because of that, it's possible for an honest party to be elected to the subcommittee of a node but be unaware of it because it wasn't in the core set of the committee for that node. We refer to such parties as oblivious honest parties. In our analysis, we assume that the bad nodes in a committee are filled only with Byzantine parties. Since oblivious parties only come from bad nodes, and their impact is that of crashed parties, we assume they are counted with the Byzantine parties coming from bad nodes and do not count them separately.

In this case, clearly the honest parties in M learn the correct information about the contents of node B', and vice versa. Every party in B polls the parties in $\ln^8 n$ nodes in $ms(B)$ to learn the parties in B'. In expectation, a party p_i in B learns the correct contents of B' from $(1 - \frac{813\ell+400}{\ln^2 n})^2$ fraction of the nodes it polls in $ms(B)$. Using a standard Chernoff bound, we find that w.h.p., more than half of the nodes polled by p_i are good leaves whose corresponding leaf in $ms(B')$ is also a good leaf, and p_i learns the correct list of parties in B'. The same goes for a party p_j in B' learning the correct parties in B. Because there are $O(n^2)$ total elections and each one involves polylogarithmically many nodes, this holds for high probability for every pair of good nodes for every election to such a node A in the network. It follows that $\sigma_i[s_j] = p_j$ and $\sigma_j[s_i] = p_i$ for any two such parties in good nodes in A's committee, where s_j and s_i are the slots corresponding to p_j and p_i, respectively, in A's committee. The remainder of the lemma follows from the fact that at most $\frac{814}{\ln^2 n}$ nodes B in A's commitee are bad nodes, and that each of the good nodes contains at least $(f_{\ell-2} - \frac{404(\ell-2)+816-404\ell}{\ln^2 n})\ln^8 n$ honest parties, which combined makes up more than $\frac{2}{3}$ of the slots in the entire committee for sufficiently large n for some $\varepsilon = O(\frac{1}{\ln \ln n})$.

Lemma 20. *If A is a layer-ℓ node with at most $\frac{814}{\ln^2 n}$ fraction bad nodes in its committee, then with high probability at least $(f_{\ell-1} - \frac{814}{\ln^2 n} - \frac{404(\ell-1)+1}{\ln^2 n} - \frac{1}{\ln^2 n})\ln^8 n$ honest parties are elected to A's subcommittee.*

Proof. By Lemma 19, the non-silent honest parties from the good nodes in A's committee form a core set, $cs_G(A)$ with high probability. It must be the case that $cs_G(A) \geq (f_{\ell-1} - \frac{814}{\ln^2 n} - \frac{404(\ell-1)+1}{\ln^2 n})c$ by the definition of a good layer $\ell - 1$ node, where $c = \Omega(\ln^{19} n)$ is the total number of slots in the commitee of A. From Lemma 8, it follows that at least $\frac{|cs_G(A)|(1-\frac{1}{\ln^2 n})\ln^8 n}{c}$ parties from $cs_G(A)$ are elected with high probability. The lemma statement follows for sufficiently large n for some $\varepsilon = O(\frac{1}{\ln \ln n})$ and due to Lemma 9.

The proof of the following lemma may be found in the full version of the paper.

Lemma 21. *The following two conditions hold for all layers ℓ s.t. $0 < \ell < \ell^\star - 1$ with high probability:*

1. *At most $\frac{813}{\ln^2 n}$ fraction of the layer ℓ nodes are bad.*
2. *The fraction of bad leaves for layer ℓ is at most $\frac{400+813\ell}{\ln^2 n}$ after the elections to layer ℓ .*

Lemma 22. *The protocol elects a supreme committee of $O(\ln^8 n)$ parties, of which more than $\frac{2}{3}$ are honest parties, that is known by all but $o(1)$ fraction of the parties with high probability.*

Proof. By Lemma 21, at most $\frac{813}{\ln^2 n}$ fraction of the layer $\ell^\star - 1$ nodes are bad, and the fraction of bad leaves for layer $\ell^\star - 1$ is $\frac{813(\ell^\star-1)+400}{\ln^2 n}$. By Lemma 20 and for some $\varepsilon = O(\frac{1}{\ln \ln n})$, it follows that more than $\frac{2}{3} \cdot \ln^8 n$ honest parties, and at most $\ln^8 n$ parties, are elected to the supreme committee with high probability. Since there is no silencing after layer $\ell^\star - 4$, the single node on layer $\ell^\star$ must be a good node, and the fraction of bad leaves for layer $\ell^\star$ must be the same as the fraction of bad leaves for layer $\ell^\star - 1$. Thus, the fraction of bad leaves for layer $\ell^\star$ is at most $\frac{813(\ell^\star-1)+400}{\ln^2 n}$. By Lemma 18, the correct list of parties in the supreme committee is passed down to all of the good leaf nodes for layer ℓ^*, from which the lemma follows.

Lemma 23. *The number of messages sent and received by each honest party, as well as the size of each message, is polylogarithmic in n.*

Proof. First, we analyse the communication done by parties as a part of monitoring sets. A layer 0 node is in a single communication tree rooted at layer ℓ of the network, for $0 \leq \ell \leq \ell^\star$. Each communication tree rooted at layer ℓ participates in SendMsgDownTree for at most $\ln^6 n$ elections on layer $\ell + 1$ for $0 < \ell < \ell^\star - 1$. It follows that a single layer 0 node is in at most $\ln^6 n$ monitoring sets for each layer of the network. Monitoring sets are involved in communication in LearnProcessors and in ConfirmElectionWinner. In a single invocation of LearnProcessors, all parties in a node in the monitoring set of one node send a list of polylogarithmically many parties to all of the parties in a single other node. For a given election, a single layer 0 node is involved in polylogarithmically many invocations of LearnProcessors. When a party in a layer 0 node responds to polling parties during LearnProcessors, it responds to at most $\ln^8 n$ parties that it believes are in that node. From Lemma 15, the fact that there are $O(\frac{\ln n}{\ln \ln n})$ layers, and the fact that each party is in at most $\ln^6 n$

layer 0 nodes, it follows that there is polylogarithmic per process communication complexity related to communication on behalf of the monitoring sets.

Next, we analyse communication incurred by SendMsgDownTree. Each party is silent in all except 10 slots that it occupies for layers ℓ, where $0 < \ell < \ell^\star - 4$. It follows that a party is not silent for at most polylogarithmically many slots per layer (for all layers of the network). As noted earlier, each communication tree rooted at layer ℓ participates in SendMsgDownTree for at most $\ln^6 n$ elections on layer $\ell + 1$ for $0 < \ell < \ell^\star - 1$. In SendMsgDownTree, all parties in a node send (for each slot they occupy in that node) a list of polylogarithmically many election winners to $O(\ln^{11} n)$ parties in $\ln^5 n$ child nodes in the communication tree. It follows that all invocations of SendMsgDownTree combined incur polylogarithmic per-process communication complexity.

Finally, we analyse communication incurred by parties as a part of the committees participating in elections. For each invocation of LightestBins, the participating parties incur polylogarithmic per-process communication complexity for BroadcastCS for polylogarithmically many slots in the committee (note that there are polylogarithmically many slots on layer $\ell^\star - 1$). For each slot occupied by a party in a committee, it sends a polling message in LearnProcessors to the parties in $\ln^8 n$ nodes in the monitoring set. Given that a party goes silent for all except 10 slots that it is elected to on a given layer, and a single node is in the committee of at most $\ln^6 n$ nodes on the next layer, the parties incur polylogarithmic per process communication complexity for communication among the parties in the committees for each election.

Lemma 24. *The* SupremeCommitteeElection *protocol completes in* polylog(n) *rounds.*

Proof. Each layer of the network consists of the simultaneous invocation for all nodes on that layer of LearnProcessors, followed by SubcommitteeElection, followed by SendMsgDownTree. Each of these protocols requires polylog in $n$rounds because the layered network has $O(\frac{\ln n}{\ln \ln n})$ and the fact that every committee contains polylog in n slots.

The lemma therefore follows from the fact that there are $O(\frac{\ln n}{\ln \ln n})$ layers.

Proof (Proof of Theorem 1). From Lemma 22 a supreme committee of polylog(n) parties is elected, of which more than $\frac{2}{3}$ are honest parties. It follows from Lemma 21 that via SendMsgDownTree, a message m held by parties in the node on layer $\ell^\star$ may be learned by all but $o(1)$ of the honest parties (due to the fraction of bad leaves for layer $\ell^\star - 1$).

The theorem follows as a direct consequence of Lemma 23 and the fact that the layered network has $O(\frac{\ln n}{\ln \ln n})$ layers, as the parties in the supreme committee may run a leader election protocol, with communication and round complexity that is polynomial in the size of the supreme committee, among themselves and disseminate the result via the communication tree.

References

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

AN90. Noga Alon and Moni Naor. Coin-flipping games immune against linear-sized coalitions (extended abstract). In *31st FOCS*, pages 46–54. IEEE Computer Society Press, October 1990.

BCG21. Elette Boyle, Ran Cohen, and Aarushi Goel. Breaking the $o(\sqrt{n})$-bit barrier: Byzantine agreement with polylog bits per party. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pages 319–330, 2021.

BDH22. Gerandy Brito, Ioana Dumitriu, and Kameron Decker Harris. Spectral gap in random bipartite biregular graphs and applications. *Combinatorics, Probability and Computing*, 31(2):229–267, 2022.

BGP92. Piotr Berman, Juan A Garay, and Kenneth J Perry. Bit optimal distributed consensus. In *Computer science: research and applications*, pages 313–321. Springer, 1992.

BKLZL20. Erica Blum, Jonathan Katz, Chen-Da Liu-Zhang, and Julian Loss. Asynchronous byzantine agreement with subquadratic communication. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 353–380. Springer, Cham, November 2020.

BL85. Michael Ben-Or and Nathan Linial. Collective coin flipping, robust voting schemes and minima of banzhaf values. In *26th FOCS*, pages 408–416. IEEE Computer Society Press, October 1985.

BN93. Ravi B. Boppana and Babu O. Narayanan. The biased coin problem. In *25th ACM STOC*, pages 252–257. ACM Press, May 1993.

CKS20. Shir Cohen, Idit Keidar, and Alexander Spiegelman. Not a coincidence: Sub-quadratic asynchronous byzantine agreement whp. In *34th International Symposium on Distributed Computing*, 2020.

CL+99. Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

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

Fei99. Uriel Feige. Noncryptographic selection protocols. In *40th FOCS*, pages 142–153. IEEE Computer Society Press, October 1999.

GPS19. Yue Guo, Rafael Pass, and Elaine Shi. Synchronous, with a chance of partition tolerance. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 499–529. Springer, Cham, August 2019.

Hae94. W.H. Haemers. *Interlacing eigenvalues and graphs*, volume FEW 675 of *Research memorandum / Tilburg University, Department of Economics*. Unknown Publisher, 1994. Pagination: 19, viii.

HPZ22. Shang-En Huang, Seth Pettie, and Leqi Zhu. Byzantine agreement in polynomial time with near-optimal resilience. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 502–514, 2022.

HPZ24. Shang-En Huang, Seth Pettie, and Leqi Zhu. Byzantine agreement with optimal resilience via statistical fraud detection. *Journal of the ACM*, 71(2):1–37, 2024.

KS10. Valerie King and Jared Saia. Breaking the $O(n^2)$ bit barrier: scalable byzantine agreement with an adaptive adversary. In Andréa W. Richa and Rachid Guerraoui, editors, *29th ACM PODC*, pages 420–429. ACM, July 2010.

KSSV06a. Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *17th SODA*, pages 990–999. ACM-SIAM, January 2006.

KSSV06b. Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Towards secure and scalable computation in peer-to-peer networks. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 87–98. IEEE, 2006.

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

Mic17. Silvio Micali. Very simple and efficient byzantine agreement. In Christos H. Papadimitriou, editor, *ITCS 2017*, volume 4266, pages 6:1–6:1, 67, January 2017. LIPIcs.

ORV94. Rafail Ostrovsky, Sridhar Rajagopalan, and Umesh V. Vazirani. Simple and efficient leader election in the full information model. In *26th ACM STOC*, pages 234–242. ACM Press, May 1994.

PS17. Rafael Pass and Elaine Shi. The sleepy model of consensus. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 380–409. Springer, Cham, December 2017.

RZ98. Alexander Russell and David Zuckerman. Perfect information leader election in log*n + O(1) rounds. In *39th FOCS*, pages 576–583. IEEE Computer Society Press, November 1998.