Round Complexity of Byzantine Agreement, Revisited[*]

T-H. Hubert Chan
HKU

Rafael Pass
CornellTech

Elaine Shi
Cornell

Abstract

Although Byzantine Agreement (BA) has been studied for three decades, perhaps somewhat surprisingly, there still exist significant gaps in our understanding regarding its round complexity. First, although expected constant-round protocols are known in the honest majority setting, it is unclear whether one has to settle for *expected* constant-round or whether there exist better protocols that are *worst-case* constant-round. Second, for the corrupt majority setting, the existence of sublinear-round BA protocols continues to ellude us except for the narrow regime when only sublinearly more than a half are corrupt.

In this paper, we make a couple important steps forward in bridging this gap. We show two main results:

1. No (even randomized) protocol that completes in *worst-case* $o(\log(1/\delta)/\log\log(1/\delta))$ rounds can achieve BA with $1-\delta$ probability, even when only 1% of the nodes are corrupt. In comparison, known expected constant-round, honest-majority protocols complete in $O(\log(1/\delta))$ rounds in the worst-case. Therefore, our lower bound is tight upto a $\log\log$ factor for the honest majority setting.
2. There exists a corrupt-majority BA protocol that terminates in $O(\log(1/\delta)/\epsilon)$ rounds in the worst case and tolerates $(1-\epsilon)$ fraction of corrupt nodes. Our upper bound is optimal upto a logarithmic factor in light of the elegant $\Omega(1/\epsilon)$ lower bound by Garay et al. (FOCS'07).

[*]A subset of the work was done while the authors were consulting for Thunder Research.

1 Introduction

A central abstraction in distributed systems and cryptography is Byzantine Agreement (BA), where a designated sender aims to communicate a bit to multiple receivers. We require two security properties, *consistency* and *validity*. Consistency requires that all honest nodes output the same bit; and validity requires that they all output the sender's bit if the sender is honest. Since the beginning of distributed computing, a foundational and important question is the *round complexity* of Byzantine Agreement. A line of elegant works have investigated this question. The celebrated work of Dolev and Strong [6] showed that there exists an $(f+1)$-round BA protocol that tolerates upto f Byzantine corruptions for any $f < n$. Further, they showed that $f+1$ rounds is optimal for *deterministic* protocols. Subsequently, it was shown that *randomized* protocols can overcome this $f+1$ round complexity lower bound. Notably, a sequence of works [3, 7, 11], beginning with Feldman and Micali [7]'s ingenious work, showed the existence of *expected constant-round* protocols in the *honest-majority* setting.

Perhaps somewhat surprisingly, despite three decades of research, significant gaps still exist in our understanding regarding the round complexity of BA. Based on the state of the affairs, two very natural and important questions are the following:

1. Do we have to settle for *expected* constant-round protocols in the honest majority setting? Can we further improve the round complexity and obtain *worst-case* constant-round protocols?

2. In the corrupt majority setting, can we design randomized BA protocols that reach agreement in *sublinear* number of rounds?

To the best of our knowledge no work has provided an answer for first question. All known expected constant-round protocols [2, 3, 7, 11] incur $O(\log(1/\delta))$ rounds in the worst case for the protocol's failure probability to be bounded by δ. Typically we require δ to be negligibly small in some security parameter κ and thus all known expected constant-round protocols incur super-logarithmic (in κ) number of rounds. So far the only known round complexity lower bound for randomized BA is due to Garay et al. [9] — they show any (even randomized) protocol that achieves BA must incur at least $\Omega(n/(n-f))$ rounds where n denotes the total number of nodes and f denotes the number of corrupt nodes. Unfortunately for the honest majority setting, their result gives rise to a constant-round lower bound which neither confirms nor rejects the existence of worst-case constant round protocols.

A couple works have partially explored the second question and provided a positive answer but only in a very limited parameter regime. Specifically, Fitzi and Nielsen [8] showed that if the number of corrupt nodes is $n/2 + k$, then an $O(k)$-round (randomized) BA protocol exists assuming the existence of a PKI and secure signatures (and their work improves the earlier result by Garay et al. [9]). In other words, so far we only know how to construct sublinear-round BA protocols in the corrupt majority setting when the number of corruptions is $n/2 + o(n)$, i.e., not too much more than a half.

1.1 Our Results and Contributions

In this paper, we revisit the round complexity of Byzantine Agreement, and make an endeavor to address the long-standing gaps in our understanding. We advance the theoretical state of the art with two novel results as explained below.

Main result 1: a new round complexity lower bound. We answer the first of the above questions negatively by proving a new round complexity lower bound. Concretely, we show that even when only 1% (or an arbitrarily small constant fraction) of the nodes are corrupt, no (even randomized) protocol that terminates in worst-case $o\left(\frac{\log(1/\delta)}{\log\log(1/\delta)}\right)$ rounds can achieve BA with $1-\delta$ probability. In comparison, known expected constant-round, honest-majority protocols [2, 3, 7, 11] incur $O(\log(1/\delta))$ rounds in the worst case. Thus *for the honest majority setting, our lower bound is tight upto a* $\log\log$ *factor*; moreover, we stress that previously, it was not known how to prove even a super-constant lower bound for this regime.

More formally, our first main theorem is the followng.

Theorem 1 (Near-logarithmic worst-case round complexity is necessary)**.** *Let $f \geq 1$ and $n \geq f+2$. Suppose that $R \leq f$ and $\frac{2nR}{f} \cdot (n+0.5) \cdot (2n/\lfloor f/R \rfloor + 1)^{R-1} \cdot \delta < 1$. Then, no (possibly randomized) protocol that terminates in R rounds (with probability 1) can achieve Byzantine Agreement with $1-\delta$ probability among n nodes in the presence of f corruptions. Furthermore, the lower bound holds even when assuming that the adversary is constrained to static Byzantine corruptions, and under any reasonable setup assumptions such as the existence of a public-key infrastructure (PKI), random oracles (RO), the ability for honest nodes to erase secrets from memory, and cryptographic hardness assumptions.*

In other words, if $\frac{f}{n} = \Theta(1)$ is some constant and $\delta \leq \frac{1}{\mathsf{poly}(n)}$, then the number R of rounds has to be $\Omega(\frac{\log\frac{1}{\delta}}{\log\log\frac{1}{\delta}})$.

Main result 2: nearly round-optimal BA protocols in the corrupt majority setting. We next show a new positive result in the corrupt majority setting. Assuming the existence of a public-key infrastructure and standard cryptographic assumptions. For any $0 < \epsilon, \delta < 1$, suppose that the adversary corrupts at most $(1-\epsilon)n$ nodes and runs in time polynomial in some security parameter κ, then we can construct a BA protocol that reaches agreement in $O(\log(1/\delta)/\epsilon)$ number of rounds with probability $1-\delta-\mathsf{negl}(\kappa)$ where $\mathsf{negl}(\kappa)$ is a negligibly small function in κ corresponding to the probability that the cryptographic primitives employed are broken. Note that in light of the elegant $\Omega(1/\epsilon)$-round lower bound by Garay et al. [9], our upper bound is nearly optimal except for a $\log(1/\delta)$ factor. Note that our result allows for ϵ to be a function in n. For example, for $\epsilon = O(1)$, we give an $O(\log(1/\delta))$-round protocol; and for $\epsilon = O(1/\sqrt{n})$, we give an $O(\sqrt{n} \cdot \log(1/\delta))$-round protocol. Finally, for any $f \leq n - \omega(\log(1/\delta))$, we acheive sublinear (in n) number of rounds which is asymptotically better than the celebrated Dolev-Strong protocol [6].

Theorem 2 (Nearly round-optimal protocols for corrupt majority)**.** *Assume the existence of a public-key infrastructure (PKI) and standard cryptographic assumptions. For parameters $\epsilon, \delta \in (0,1)$ which are allowed to be functions in n, there exists a protocol that terminates in $O(\log(1/\delta)/\epsilon)$ number of rounds and achieves BA with $1-\delta-\mathsf{negl}(\kappa)$ probability in the presence of an adversary that adaptively corrupts at most $(1-\epsilon)n$ nodes and runs in time polynomial in κ.*

We stress that previously, except for the narrow parameter regime $f = 0.5n + o(n)$, no sublinear-round protocol is known for the corrupt majority setting, not even under *static* corruption and making any conceivable assumption including very strong ones such as random oracles and the ability of honest nodes to erase secrets from memory. Our protocol works in a model where the adversary may adaptively corrupt nodes in the middle of the execution, as long as the adversary cannot retroactively erase messages that were already sent before the corruption took place.

2 Preliminaries

2.1 Protocol Execution Model

We assume a standard protocol execution model with n nodes indexed with $[n] := \{1, 2, \ldots, n\}$. An external party called the environment and denoted $\mathcal{Z}$ provides inputs to honest nodes and receives outputs from the honest nodes. An adversary denoted $\mathcal{A}$ controls a subset of the nodes which are said to be *corrupt*; all other nodes are said to be *honest*. All corrupt nodes are under the control of $\mathcal{A}$, i.e., the messages they receive are forwarded to $\mathcal{A}$, and $\mathcal{A}$ controls what messages they will send once they become corrupt. The adversary $\mathcal{A}$ and the environment $\mathcal{Z}$ are allowed to freely exchange messages any time during the execution. To capture protocols that employ cryptography, we assume that all nodes as well as $\mathcal{A}$ and $\mathcal{Z}$ are Interactive Turing Machines that run in time polynomial in some security parameter κ; further, we assume that κ is known to all nodes as well as $\mathcal{A}$ and $\mathcal{Z}$.

We assume a standard *synchronous* network model. Whenever honest nodes send a message, the message is delivered to honest recipients at the beginning of the next round. For our lower bound, we shall assume a *static* adversary that corrupts nodes in advance before the start of the execution. For our upper bound, we shall assume an *adaptive* adversary that can corrupt nodes in the middle of the execution. Note that this makes our lower and upper bounds both stronger. In the adaptive corruption case, we shall assume that the adversary can observe all currently honest nodes' messages in round r before deciding which subset of these nodes to corrupt in round r. Suppose an honest node P sends a message in some round r and then becomes corrupt in the same round — in this case we assume that the adversary cannot perform "after-the-fact" removal and a-posteriori delete the message that was sent by P in round r before it became corrupt. However, now that P is corrupt, the adversary may now inject additional round-r messages on behalf of P.

2.2 Byzantine Agreement

In this section we formally define Byzantine Agreement. Recall that there are n nodes numbered $1, 2, \ldots, n$. Without loss of generality, we shall assume that node 1 is the designated sender.

Syntax. Prior to protocol start, the sender receives an input $b \in \{0, 1\}$ from the environment $\mathcal{Z}$. At the end of the protocol, every node i (including the sender) outputs a bit b_i to the environment $\mathcal{Z}$.

Security definition. We say that a protocol (satisfying the above syntax) achieves BA with probability p with respect to $(\mathcal{A}, \mathcal{Z})$, iff with probability at least p over the choice of the randomized execution, the following properties are satisfied:

- *Consistency.* If an honest honest node outputs b_i and another honest node outputs b_j to $\mathcal{Z}$, then it must hold that $b_i = b_j$.
- *Validity.* If the designated sender remains honest throughout and its input is b, then any honest node's output to $\mathcal{Z}$ must be b.

We say that a protocol terminates in *worst-case* R rounds iff it terminates in R rounds with probability 1.

3 Near-Logarithmic Worst-Case Round Complexity is Necessary

In this section, we will prove Theorem 1. For convenience, we will use a variant of the BA definition in our lower bound proof. In this variant, instead of having a designated sender, every node receives an input bit. The *consistency* requirement is the same as before; for *validity*, we require that if all nodes are honest and they all receive the same input b, then all nodes should output b. The literature also refers to this variant as "agreement" and the variant defined earlier in Section 2.2 as "broadcast". It is not hard to see that if there exists a worst-case R-round broadcast protocol Π_{bcast} secure against f Byzantine corruptions, then we can construct an agreement protocol Π_{agree} that completes in worst-case R rounds and secure against the same number of corruptions. Basically, Π_{agree} runs n instances of Π_{bcast} where in instance $i \in [n]$, node i is the designated sender and tries to broadcast its input bit to everyone. At the end of R rounds, a node outputs 0 if all Π_{bcast} instances output 0. Else, it outputs 1.

3.1 Warmup: $f+1$ Round Complexity Lower Bound for Deterministic Protocols

We begin by reviewing the famous $f + 1$ round complexity lower bound for deterministic protocols. We will present a re-exposition [1] of Dolev and Strong's proof [6] that is conducive to our proof for randomized protocols later. Since we are concerned about the round complexity of the protocol, without loss of generality, we may assume that in every round, everyone sends a message to everyone.

Consider a deterministic protocol that terminates in f rounds where f is the number of corrupt nodes. To show that such a protocol cannot achieve BA, we will consider a sequence of executions. In each execution, some nodes may drop a subset to all of the messages they ought to send in some rounds but all nodes otherwise follow the honest protocol. Anyone who ever drops a message is corrupt and everyone else is honest. The idea is to show that there exists a sequence of executions beginning at the all-honest execution where every node receives the input bit 0, and ending at the all-honest execution where every node receives the input bit 1; furthermore, the following two invariants must be respected:

1. *Neighboring constraint:* for any pair of adjacent executions, there exists at least one node that is honest in both executions, and moreover its view has not changed in between these executions.

2. *Corruption constraint:* in every execution in the sequence, in each round no more than one node's outgoing messages may be dropped — since the protocol has f rounds, the total number of corrupt nodes in any execution is at most f.

Now, assume this f-round protocol achieves BA, then we can reach a contradiction in the following way. In the first all-honest, all-0-input execution, by validity, every node should output 0. Now, due to the neighboring constraint, there must be a node P that is honest in both the first and the second executions, and its view is identical in both executions. Now P must output 0 too in the second execution. Recall that every execution has at most f corrupt nodes, and thus by consistency, all honest nodes must output 0 in the second execution. We can now apply this reasoning inductively. Suppose that all honest nodes output 0 in the i-th execution, we can show that due to the neighboring and the corruption constraints, all honest nodes must output 0 in the $(i+1)$-st execution too. We thus conclude that all honest nodes must output 1 in the final execution, which is an all-honest, all-1-input execution; but this violates validity.

Therefore, the crux of the proof is how to construct a sequence of executions satisfying the above requirements. Dolev and Strong [6] show that indeed this can be accomplished by dropping or restoring one message pertaining to one round at a time in a carefully crafted fashion. In Appendix C, we will present a concrete example for the special case $f = 2$, i.e., no 2-round deterministic protocol can realize BA in the presence of 2 corruptions.

3.2 Our Randomized Lower Bound

To prove a near-logarithmic lower bound for randomized protocols, we adopt a similar idea to go through a sequence of executions, starting at the all-honest all-0-input execution and ending at the all-honest all-1-input execution. One crucial difference is that now each execution is *randomized*, and this raises several new challenges for our proof.

First, we modify the aforementioned two invariants as follows:

1. *Neighboring constraint:* for every pair of adjacent executions, there exists at least one node P that is honest in both executions, such that P's view is *identically distributed* in both executions.
2. *Corruption constraint:* in every execution, some nodes may drop a subset to all of the messages they are supposed to send — we require that the total number of nodes that drop messages is upper bounded by f.

Recall that earlier in the proof for deterministic protocols [6], it suffices to show that such a sequence of executions *exists*. When the executions are randomized, showing existence is no longer sufficient. We need to show a stronger statement, i.e., there is a relatively *short* sequence of executions satisfying the aforementioned requirements — specifically, as we argue below, the probability that all honest nodes output 0 will degrade with each execution.

More concretely, suppose that there is an R-round randomized protocol that achieves BA where R satisfies the constraints in the theorem statement. If we can indeed construct a sequence of M executions satisfying the aforementioned requirements for a sufficiently small M, we can now reach a contradiction as follows. By validity, we have that in the first all-honest all-0-input execution, all honest nodes must output 0 with $1-\delta$ probability. Now, due to the neighboring constraint, if all honest nodes output 0 with probability $1-\nu$ in execution $(i-1)$, then some honest node (whose distribution on view has not changed between executions $i-1$ and i) must output 0 with probability $1-\nu$ in execution i too. Now, by the consistency requirement of execution i, we may conclude that every honest node outputs 0 with probability at least $1-\nu-\delta$ in execution i. In this way we may conclude that the probability that all honest nodes output 0 in the final, all-honest all-1-input execution is lower bounded by $1-M\delta$ where M denotes the total number of executions in this sequence. Therefore as long as $1-M\delta>\delta$, we violate the validity requirement of the final execution thus leading to a contradiction.

Now the biggest challenge is how to construct a short sequence of executions satisfying the above invariants. A naïve approach, obviously, is to adopt the same sequence in the known deterministic proof (see earlier works [1, 6] and Appendix C). Since this particular sequence drops or restores one message at a time, it results in a long sequence of length $O\left((C\cdot n)^R\right)$ where C is a suitable constant. To satisfy the relation $1-M\delta>\delta$, R must be $O(\log(1/\delta)/\log n)$. For the typical parameter regime where n is a polynomial function in κ and δ is negligibly small in κ, this results in a super-constant (in κ) lower bound assuming that the number of corrupt nodes f is also super-constant. Note that even this simple super-constant lower bound is interesting since it already rules out the existence of worst-case constant-round protocols that achieve negligibly small in κ failure probability.

Our goal, however, is to prove a stronger lower bound. To see our key insight, consider the typical case when $f=\Theta(n)$ and we would like to rule out $R=o\left(\log(1/\delta)/\log\log(1/\delta)\right)$-round protocols. Since R is relatively small w.r.t. f, in each execution, we can afford to have more nodes dropping messages in each round. Our proof therefore adopts a *batching* idea to shorten the sequence of executions. First, we allow for upto $B:=\lfloor f/R\rfloor$ nodes to drop messages per round such that the corruption constraint is respected. Now, instead of altering one message at a time, we alter a may batch of messages from B sources to B destinations at a time for any non-final round. However, for technical reasons that will become obvious later, in the final round we still need to drop or restore one message at a time — this turns out to be necessary for satisfying the neighboring constraint for the final round.

3.2.1 Terminology and Notations

We begin by defining some useful notations.

Batch. Let $B := \lfloor f/R \rfloor \geq 1$ denote the batch size. We may divide the nodes n into $m = \lceil n/B \rceil$ batches. We use the notation $\langle i \rangle$ to denote the i-th batch, that is

$$\langle i \rangle := \begin{cases} \{(i-1)B+1, (i-1)B+2, \ldots, iB\} & \text{if } i \in [m-1] \\ \{(m-1)B+1, (m-1)B+2, \ldots, n\} & \text{if } i = m \end{cases}$$

Representing an execution as a communication graph. Henceforth we use the notation $i \xrightarrow{r} j$ to denote the r-th round message from node i to node j where $r \in [R]$ and $i, j \in [n]$. Similarly, for $S, S' \subseteq [n]$, $S \xrightarrow{r} S'$ denotes the set of all round-r messages from any node in S to any node in S'. For convenience, we often use $*$ to denote wildcard, i.e., $* := [n]$. We often abuse the notation $j \in [n]$ to denote a singleton set $\{j\}$ whenever convenient.

For convenience, we shall henceforth describe each execution as a *communication graph* of R rounds, *tagged with a set of input bits for all nodes.* A message $i \xrightarrow{r} j$ is called an *edge* of the communication graph. Therefore, an all-honest execution would result in a *complete graph* with the entire edge set $\{[n] \xrightarrow{r} [n]\}_{r \in [R]}$. If an edge $i \xrightarrow{r} j$ is missing from the graph, it means that i drops the message it ought to send to j in the r-th round.

3.2.2 Recursive Algorithm for Defining a Sequence of Executions

We now define a recursive algorithm for specifying the sequence of executions in Figure 1. The entire sequence of executions is defined by the procedure **Main**, which invokes the recursively defined algorithms **Delete** and **Restore**. All algorithms modify a global communication graph G. Initially, when **Main** is invoked, this graph G is initialized to be a complete communication graph of R round with all-0 inputs.

Intended functionalities. We state the intended functionalities of the algorithms and later we will prove correctness in Facts 1 and 2.

- Suppose that currently the graph G has a complete set of edges in any round $r' \geq r$ (henceforth this is said to be the *input assumption* for **Delete**). Then, **Delete**$(\langle i \rangle, r)$ removes from G all edges $\langle i \rangle \xrightarrow{r'} *$ for any round $r' \geq r$. The sequence of intermediate graphs encountered during the process are output.

- Suppose that in the current graph G, all edges $\{\langle i \rangle \xrightarrow{r'} *\}_{r' \geq r}$ are missing but all other edges in round $r' \geq r$ are complete (henceforth this is said to be the *input assumption* for **Restore**). Then, **Restore**$(\langle i \rangle, r)$ restores in G all edges $\langle i \rangle \xrightarrow{r'} *$ for any $r' \geq r$. The sequence of intermediate graphs encountered during the process are output.

- If the **Main** function is invoked for the complete R-round communication graph with all-0 input, then the final graph will be the complete R-round communication graph with all-1 input. All intermediate graphs encountered along the way will be output.

The following facts state the correctness of the algorithms in Figure 1 and are easy to verify.

```
Delete(⟨i⟩, r):

  If $r = R$:
      For $j \in [n]$: DeleteOne($\langle i\rangle, j, R$)

  Else:
      For $j \in [m]$:
          – Delete($\langle j\rangle, r+1$)
          – DeleteOne($\langle i\rangle, \langle j\rangle, r$)
          – Restore($\langle j\rangle, r+1$)
      Delete($\langle i\rangle, r+1$)
```

```
Restore(⟨i⟩, r):

  If $r = R$:
      For $j \in [n]$: RestoreOne($\langle i\rangle, j, R$)

  Else:
      Restore($\langle i\rangle, r+1$)
      For $j \in [m]$:
          – Delete($\langle j\rangle, r+1$)
          – RestoreOne($\langle i\rangle, \langle j\rangle, r$)
          – Restore($\langle j\rangle, r+1$)
```

```
Main():
  For $i \in [m]$:
      – Delete($\langle i\rangle, 1$)
      – Flip input bits for $\langle i\rangle$ and output
      – Restore($\langle i\rangle, 1$)
```

```
DeleteOne($S, S', r$):
  Delete all edges $S \xrightarrow{r} S'$ and output

RestoreOne($S, S', r$):
  Restore all edges $S \xrightarrow{r} S'$ and output
```

Figure 1: **Recursive algorithm for defining a sequence of executions.** All algorithms modify a global graph. Initially, when **Main** is invoked, the graph is a complete communication graph of R rounds with all-0 inputs. Whenever **output** is called, the current graph's state is output as the next execution in the sequence.

Fact 1. *The* **Delete** *and* **Restore** *algorithms are correct in the following sense:*

1. *they realize the intended functionalities stated earlier in Section 3.2.2; and*
2. *if* **Delete** *or* **Restore** *is invoked upon a graph that satisfies its respective input assumption, then all recursive calls it makes to* **Delete** *and* **Restore** *will satisfy the respective input assumptions.*

Proof. Straightforward through line-by-line mechanical verification of the pseudocode and through induction on the parameter r. □

Fact 2. *Suppose that the* **Main** *function is invoked for the complete communication graph with all-0 input, and that the* **Delete** *and* **Restore** *functions are correct as defined in Fact 1. Then, the calls to* **Delete** *and* **Restore** *made by* **Main** *satisfy the respective input assumptions. Furthermore, the final graph G is the complete communication graph with all-1 input.*

Proof. Straightforward. □

3.2.3 Analysis

Henceforth we always assume that the **Main** function is invoked for the complete communication graph with all-0 input. We first prove that the sequence of graphs output respect the neighboring and the corruption constraints defined earlier.

Lemma 1 (Corruption constraint)**.** *The sequence of graphs output respect the corruption constraint.*

Proof. **Last round:** *DeleteOne*$(\langle i \rangle, j, R)$ can only be invoked from within **Delete**$(\langle i \rangle, R)$. By Facts 1 and 2, when *DeleteOne*$(\langle i \rangle, j, R)$ must be invoked upon a graph G whose last-round edges are all complete. Therefore, at any time, at most B nodes can have last-round outgoing messages missing.

All other rounds $r < R$: *DeleteOne*$(\langle i \rangle, \langle j \rangle, r)$ can only be invoked from within **Delete**$(\langle i \rangle, r)$. *DeleteOne*$(\langle i \rangle, \langle j \rangle, r)$ must be invoked upon a graph G whose edges in rounds r and greater are all complete, except that outgoing edges from $\langle j \rangle$ in rounds $r+1$ and greater are all missing. Therefore, at any time, at most B nodes can have round-r outgoing messages missing. ☐

Lemma 2 (Neighboring constraint)**.** *The sequence of graphs output respect the neighboring constraint.*

Proof. **Last round:** *DeleteOne*$(\langle i \rangle, j, R)$ can only be invoked from within **Delete**$(\langle i \rangle, R)$. By Facts 1 and 2, when *DeleteOne*(i, R) must be invoked upon a graph G whose last-round edges are all complete. Now, each call to *DeleteOne*$(\langle i \rangle, j, R)$ where $j \in [n]$ removes the edges $\langle i \rangle \xrightarrow{R} j$. Due to Lemma 1 and the facts that $B = \lfloor f/R \rfloor$ and $n \geq f+2$, every time a last-round edge is removed in this way, there must exist at least one remaining honest node P whose view is completely unaffected, i.e., P's view is identically distributed before and after the removal of this last-round edge.

Similarly, *RestoreOne*$(\langle i \rangle, j, R)$, can only be invoked upon a graph G whose last-round edges are all complete except that the last-round edges from $\langle i \rangle$ are missing. Each call to *DeleteOne*$(\langle i \rangle, j, R)$ where $j \in [n]$ restores the edges $\langle i \rangle \xrightarrow{R} j$. Due to Lemma 1 and the facts that $B = \lfloor f/R \rfloor$ and $n \geq f+2$, every time a last-round edge is restored in this way, there must exist at least one remaining honest node P whose view is completely unaffected.

All other rounds $r < R$: *DeleteOne*$(\langle i \rangle, \langle j \rangle, r)$ can only be invoked from within **Delete**$(\langle i \rangle, r)$. The line **Delete**$(\langle j \rangle, r+1)$ deletes all edges in rounds $r+1$ or greater outgoing from $\langle j \rangle$. At this moment, *DeleteOne*$(\langle i \rangle, \langle j \rangle, R)$ obviously does not affect any honest nodes' view. Similarly, *RestoreOne*$(\langle i \rangle, \langle j \rangle, r)$ can only be invoked inside **Restore**$(\langle i \rangle, r)$. The line **Delete**$(\langle j \rangle, r+1)$ deletes all edges in rounds $r+1$ or greater outgoing from $\langle j \rangle$. At this moment, *RestoreOne*$(\langle i \rangle, \langle j \rangle, R)$ obviously does not affect any honest nodes' view.

Inside **Main**, the line **Delete**$(\langle i \rangle, 1)$ deletes all edges outgoing from $\langle i \rangle$. Obviously at this moment, changing $\langle i \rangle$'s input bits do not affect any honest nodes' view. ☐

Lemma 3 (Number of executions)**.** *A call to* **Main**() *for a complete R-round communication graph with all-0 inputs would output at most* $M = 2m \cdot (\frac{(2m+1)^{R-1}-1}{2} + n \cdot (2m+1)^{R-1})$ *graphs.*

Proof. Let $T_{\text{delete}}(d)$ denote the number of graphs output by a call to **Delete**$(\_, R-d+1)$, and let $T_{\text{restore}}(d)$ denote the number of graphs output by a call to **Restore**$(\_, R-d+1)$. We thus have the following recurrence:

$$T_{\text{delete}}(d) = m \cdot (T_{\text{delete}}(d-1) + T_{\text{restore}}(d-1) + 1) + T_{\text{delete}}(d-1), \quad T_{\text{delete}}(1) = n$$

$$T_{\text{restore}}(d) = m \cdot (T_{\text{delete}}(d-1) + T_{\text{restore}}(d-1) + 1) + T_{\text{restore}}(d-1), \quad T_{\text{restore}}(1) = n$$

Solving the recurrence, we have that

$$T_{\text{delete}}(d) = T_{\text{restore}}(d) = \frac{(2m+1)^{d-1}-1}{2} + n \cdot (2m+1)^{d-1}$$

The bound follows because the **Main**() function consists of m calls to **Delete**$(\langle * \rangle, 1)$, followed by m calls to **Restore**$(\langle * \rangle, 1)$. ☐

Proof of Theorem 1. We are now ready to use Lemmas 1, 2 and 3 to prove the lower bound result.

Lemma 1 says that in each execution graph produced, there are at most f corrupted nodes. Lemma 2 states that between adjacent execution graphs, there exists at least one honest node whose (distribution of) view does not change. Finally, Lemma 3 gives the number M of execution graphs produced. Therefore, as argued earlier in our intuition explanation, if the probability of validity is at least $1 - \delta$ for each execution graph, and $(M + 1)\delta < 1$, then the first execution graph (with all inputs 0) and the last execution graph (with all inputs 1) will cause the desired contradiction.

4 Nearly Round-Optimal BA for Corrupt Majority

We now proceed to describe our upper bound for the corrupt majority setting and thus prove Theorem 2.

4.1 Warmup: Any Constant Fraction of Static Corruption

For simplicity, let us first focus on the simpler case when the adversary is constrained to making static corruptions. Let ϵ denote the fraction of honest nodes, where ϵ can potentially be a function of n; however, as a warmup, we assume ϵ to be some arbitrarily small constant in this section. We will later extend our approach to more general choices of ϵ and to the case of adaptive corruptions. We stress, however, that except for the narrow parameter regimes in Garay et al.'s result [9], previously it was unknown how to achieve sublinear-round BA in the corrupt majority setting even assuming static corruptions. We use $\delta > 0$ to denote the failure probability (for consistency).

Flawed strawman approach. One tempting but flawed approach is to randomly elect a small committee of $\log(1/\delta)$ nodes — for the time being, imagine that a random leader election oracle exists — and have the committee run a corrupt-majority BA protocol such as Dolev-Strong [6]. For the special case $\epsilon = \Theta(1)$ and static corruption, it is not hard to show that except with $O(\delta)$ probability, the committee consists of at least 1 honest node. Thus all honest nodes within the small committee can reach agreement on a bit b^* in $\log(1/\delta)$ number of rounds. Unfortunately, there does not seem to be any straightforward to securely convey this bit to the non-committee nodes (note that the common approach of taking a majority vote among the committee fails to work in the corrupt majority setting).

To resolve this challenge, our insight is to combine the random committee election idea and the Dolev-Strong protocol in a non-blackbox manner.

Background: the Dolev-Strong protocol. We start by reviewing the classical Dolev-Strong protocol [6] that achieves linear round complexity and tolerates any number of corruptions — henceforth the term "multicast" means "send to everyone":

- Every node i maintains an $\mathsf{Extracted}_i$ set that is initialized to be empty. In round 0, the sender signs its input bit b, and multicasts b and the signature.
- For each round $r = 1 \ldots n$: for each bit $b \in \{0, 1\}$, if node i has observed valid signatures on b from at least r distinct nodes including the designated sender and $b \notin \mathsf{Extracted}_i$: compute a signature on b; multicast b and all signatures it has observed on b (including its own); include b in $\mathsf{Extracted}_i$.
- At the end of the protocol, each node i outputs the bit contained in $\mathsf{Extracted}_i$ if $|\mathsf{Extracted}_i| = 1$; else it outputs a canonical bit 0.

This protocol retains consistency, if the number K of rounds is strictly larger than the number f of (eventually) corrupt nodes. First, if an honest node i first adds a bit b to its $\mathsf{Extracted}_i$ set in any round $r < K$, then by the end of round $r+1$, b must be in every honest node's $\mathsf{Extracted}$ set. Further, if a honest node i first adds a bit b to its $\mathsf{Extracted}_i$ set in the last round K (which is at least $f+1$), then at least one out of the $K \geq f+1$ signatures it has observed in round K must be from an honest node — it holds that this honest node must have added b to its $\mathsf{Extracted}$ set in some round $r < K$ before the last round; and thus by the end of the last round K, every honest node will have b in its $\mathsf{Extracted}$ set.

Achieving agreement for non-committee members. Recall that the problem with the naïve committee election approach is how to convey the committee's decision to the non-committee members. To this end we will combine the committee election idea with Dolev and Strong's protocol in a non-blackbox manner. Suppose that a leader election oracle exists that helps us elect a committee of $\log(1/\delta)$ nodes after the adversary chooses the corrupt nodes. As argued earlier, except with probability $O(1/\delta)$, there is at least one honest node in the committee.

Henceforth we assume that only the committee members are authorized signers and signatures from any non-committee node will be ignored. Our key insight is to divide each round r of the Dolev-Strong protocol into two mini-rounds:

- Every node i maintains an $\mathsf{Extracted}_i$ set that is initialized to be empty. In round 0, the sender signs its input bit b, and multicasts b and the signature.
- For each round $r = 1, 2, \ldots, S+1$ where $S = \log(1/\delta)$ denotes the committee size,
  1. In the first mini-round, if *any* node i receives a bit $b \notin \mathsf{Extracted}_i$ with r signatures from distinct signers, it adds b to $\mathsf{Extracted}$ and multicasts b tagged with all signatures observed so far for b.
  2. In the second mini-round, only the committee members perform the actions above and moreover a committee member always appends its own signature for b when multicasting b (and all other seen signatures on b).
- Each node i outputs the bit contained in $\mathsf{Extracted}_i$ if $|\mathsf{Extracted}_i| = 1$; else it outputs a canonical bit 0.

Note that this approach guarantees that if any honest node newly adds a bit b to its $\mathsf{Extracted}$ set in round r, then all committee nodes must have signed it by the end of round r (if not earlier) and multicast the corresponding signature. Thus in the first mini-round of round $r+1$, every honest node will have added b to its $\mathsf{Extracted}$ set. At this moment, it is not difficult to see that as long as one committee member is honest, if the above protocol is executed for at least $f+1$ rounds then we can argue consistency using a similar approach as Dolev-Strong.

4.2 Achieving Adaptive Security and Removing the Leader Election Oracle

The above protocol enables the committee to securely convey its decision to non-committee nodes; unfortunately, the protocol does not defend against adaptive corruptions. Specifically, since the elected committee is small relative to n, an adaptive adversary can simply corrupt all committee members after they are elected. We now present an approach for achieving adaptive security borrowing the "bit-specific committee election" idea that was previously employed in the construction of small-bandwidth honest-majority BA protocols by Abraham et al. [2] and Chan et al. [4]. As a by-product we will have instantiated the leader election oracle that was needed earlier.

Our idea is to *tie the committee election to each individual bit*, i.e., there is a separate committee that are allowed to vote on 0 and 1 respectively (henceforth called the 0-committee and the 1-committee respectively), and the designated sender is in both committees. Specifically, Abraham et al. [2] and Chan et al. [4] describe how to realize bit-specific committee election using a suitable Verifiable Random Function (VRF) with adaptive security. Take $b = 0$ as an example. For a node i to determine if he is on the 0-committee, he checks the following:

$$\text{let } (\rho, \pi) := \mathsf{VRF}_{\mathsf{sk}_i}(0), \text{ and check if } \rho < D_p$$

Here sk_i is his private key, D_p is an appropriate difficulty parameter that determines the success probability (denoted p) of each election attempt, and π is a proof generated by the VRF which will be used below for verification. Concretely, the probaiblity p is chosen such that the expected number of nodes elected into either the 0-committee or 1-committee is $\log(1/\delta)$. , To convince others that i is indeed an eligible member of the 0-committee, i reveals both ρ and π, and everyone can now verify, using i's public key, that indeed ρ is the correct outcome of the VRF.

We now explain how to use bit-specific committee election to achieve adaptive security. Suppose a 0-committee member i becomes immediately corrupt after signing 0 and multicasting signatures on 0. However, corrupting node i does not necessarily help voting for the bit 1 — in particular, since the two committees are independently selected, node i is *only as good as any other node in terms of its likelihood of being elected into the* 1*-committee.* Thus, corrupting node i is only as good as corrupting any other node at this point. In our proofs in the appendices, we will formalize the above intuition.

Putting it altogether. We say that a tuple (b, i, π) is a valid *vote* on b iff either 1) $i = 1$ is the designated sender and π is a valid signature on b from i; or 2) $i \neq 1$ and π is a valid VRF proof proving i to be in the b-committee. The protocol is described below.

- Every node i initializes $\mathsf{Extracted}_i := \emptyset$. The sender signs its input bit b and multicasts b as well as the signature.
- For round $r = 1, \ldots, \log(1/\delta)$, every node i performs the following:
  1. First mini-round: for every $b \notin \mathsf{Extracted}_i$ such that the node has observed at least r votes from distinct nodes including the sender: add b to $\mathsf{Extracted}_i$; multicast b and all observed votes on b.
  2. Second mini-round: for every $b \notin \mathsf{Extracted}_i$, if node i belongs to the b-committee and moreover the node has observed at least r votes from distinct nodes including the sender: add b to $\mathsf{Extracted}_i$; compute a new vote on b; multicast b and all observed votes on b (including the newly created one).
- Every node i outputs the bit contained in $\mathsf{Extracted}_i$ if $|\mathsf{Extracted}_i| = 1$; else output a canonical bit 0.

4.3 Deferred Technical Contents

In the appendices, we shall formally present our upper bound result for the corrupt majority setting and extend it to more general choices of ϵ. We will also provide formal proofs of security.

References

[1] Lecture notes for 6.852: Distributed algorithms fall, 2009. MIT Open Courseware, `https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-852j-distributed-algorithms-fall-2009/lecture-notes/MIT6_852JF09_lec05.pdf`.

[2] Ittai Abraham, T-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In *PODC*, 2019.

[3] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. Synchronous byzantine agreement with optimal resilience, expected $o(n^2)$ communication, and expected $o(1)$ rounds. In *Financial Cryptography and Data Security (FC)*, 2019.

[4] T-H. Hubert Chan, Rafael Pass, and Elaine Shi. Consensus through herding. In *Eurocrypt*, 2019.

[5] Jing Chen and Silvio Micali. Algorand: The efficient and democratic ledger. https://arxiv.org/abs/1607.01341, 2016.

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

[7] Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous byzantine agreement. In *SIAM Journal of Computing*, 1997.

[8] Matthias Fitzi and Jesper Buus Nielsen. On the number of synchronous rounds sufficient for authenticated byzantine agreement. In *Proceedings of the 23rd International Conference on Distributed Computing*, DISC'09, pages 449–463, Berlin, Heidelberg, 2009. Springer-Verlag.

[9] Juan Garay, Jonathan Katz, Chiu-Yuen Koo, and Rafail Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 11 2007.

[10] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, June 2012.

[11] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. *J. Comput. Syst. Sci.*, 75(2):91–112, February 2009.

[12] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Crypto*, 2017.

A Formal Description of $\mathcal{F}_{\text{mine}}$-Hybrid Protocol

In the sections to follow, we will formally present our upper bound for the corrupt majority case. We will first describe our protocol assuming an idealized oracle called $\mathcal{F}_{\text{mine}}$ that is in charge of random eligibility election — this approach allows us to "abstract away" the cryptography and focus on analyzing the stochastic properties of the protocol first. Later in Section B, we will show how to leverage standard techniques to remove the $\mathcal{F}_{\text{mine}}$ assumption and instantiate it with appropriate, adaptively secure cryptography.

A.1 Ideal Functionality $\mathcal{F}_{\text{mine}}$ for Random Eligibility Determination

The idealized oracle $\mathcal{F}_{\text{mine}}$ provides the following functionality. A node i can query $\mathcal{F}_{\text{mine}}$ to check if it is an eligible member of the b-committee where $b \in \{0, 1\}$. Upon receiving such a query, $\mathcal{F}_{\text{mine}}$ flips a random coin (with appropriate probability) to determine the answer; further $\mathcal{F}_{\text{mine}}$ stores this answer and returns it to any node that queries it henceforth.

More formally, the $\mathcal{F}_{\text{mine}}$ ideal functionality has two activation points:

- Whenever a node i calls $\texttt{mine}(b)$ for the first time where $b \in \{0, 1\}$, $\mathcal{F}_{\text{mine}}$ flips a random coin (parametrized with an appropriate probability p) to decide if i is a committee member for b.

  Henceforth if a node i calls $\mathcal{F}_{\text{mine}}.\texttt{mine}(b)$, we also say that i makes a mining attempt for the bit b.

- If node i has called $\texttt{mine}(b)$ and the attempt is successful, anyone who calls $\mathcal{F}_{\text{mine}}.\texttt{verify}(b, i)$ will obtain an answer of 1; all other calls to $\mathcal{F}_{\text{mine}}.\texttt{verify}(b, i)$ will return 0.

Henceforth in the paper, we assume that the choice of the success probability p is a global, public parameter. We will describe how to choose p later.

A.2 Formal Protocol in the $\mathcal{F}_{\text{mine}}$-Hybrid World

We describe how to achieve adaptively secure BA with sublinear round complexity, tolerating $1 - \epsilon$ fraction of corruption for any arbitrarily small positive constant ϵ. Recall that without loss of generality, we assume that node 0 is the designated sender.

Valid vote. With respect to some moment in time, a *valid vote* for the value b from node i is of the following form:

- If $i = 0$, i.e., i is the sender, then a valid vote is of the form $(b, 0, \texttt{Sig}_0(b))$, where $\texttt{Sig}_0(b)$ denotes a valid signature from the sender on the bit b.
- For $i \neq 0$, a valid vote w.r.t. some time t is of the form (b, i) such that $\mathcal{F}_{\text{mine}}.\texttt{verify}(b, i)$ returns 1 at time t, i.e., by time t, node i must have called $\mathcal{F}_{\text{mine}}.\texttt{mine}(b)$ and the result must have been successful.

Valid vote batch. For $r \geq 1$, a valid r-batch of votes for value b consists of valid votes for value b from r distinct nodes, one of which must be the sender 0. Note that just like the definition of a valid vote, a valid vote batch is also defined w.r.t. to some moment of time (which we sometimes omit writing explicitly if the context is clear).

Since we have explained the intutition behind our protocol earlier, we now give a formal presentation of the protocol in Figure 2 — here "multicast" means sending a message to everyone.

A.3 Analysis in the $\mathcal{F}_{\text{mine}}$-Hybrid World

In this subsection, we shall prove the following theorem for our $\mathcal{F}_{\text{mine}}$-hybrid-world protocol described in Figure 2.

Theorem 3. *Assume that the signature scheme is secure. For any $0 < \epsilon, \delta < 1$ (that can be functions of n), the $\mathcal{F}_{\text{mine}}$-hybrid Byzantine agreement protocol described in Figure 2 satisfies consistency (with probability at least $1 - \delta$) and validity (subject to the security of the signature scheme) and terminates in $2 \cdot \lceil \frac{3}{\epsilon} \ln \frac{2}{\delta} \rceil$ rounds (with probability 1) w.r.t. any non-uniform p.p.t. $(\mathcal{A}, \mathcal{Z})$ that corrupts no more than $(1 - \epsilon)n$ nodes.*

Byzantine Agreement: Synchronous Network with Corrupt Majority

Parameters: Let ϵ be the fraction of forever honest nodes and δ be the desired failure probability. $\mathcal{F}_{\text{mine}}$ is instantiated with a probability $p := \min\{1, \frac{1}{\epsilon n} \log \frac{2}{\delta}\}$. Let $R = \lceil \frac{3}{\epsilon} \cdot \ln \frac{2}{\delta} \rceil$ be the total number of stages.

Stage 0: Initialization. No message is multicast in this stage.

- The sender 1 produces a valid 1-batch of vote for its value b_0 by producing a signature $\texttt{Sig}_1(b_1)$.
- Every node i sets $\mathsf{Extracted}_i \leftarrow \emptyset$.

Stage $r \in [1..R]$. Each such stage consists of 2 rounds.

1. In the first round, every node i performs the following:
   - For each bit b, if node i has seen a valid r-batch of votes for b and $b \notin \mathsf{Extracted}_i$, then it multicasts any such r-batch for b to everyone, and sets $\mathsf{Extracted}_i \leftarrow \mathsf{Extracted}_i \cup \{b\}$.
2. In the second round, each node $i \neq 1$ does the following. For each bit b, if it has seen a valid r-batch of votes for b and node i has never called $\mathcal{F}_{\text{mine}}.\texttt{mine}(b)$ before, then it calls $\mathcal{F}_{\text{mine}}.\texttt{mine}(b)$ and executes the following if the result is successful:
   - It sets $\mathsf{Extracted}_i \leftarrow \mathsf{Extracted}_i \cup \{b\}$.
   - It multicasts a valid $(r+1)$-batch of votes for b, possibly by adding its own valid vote (b, i).

Stage $R+1$: Termination. No message is multicast in this stage. Every node i performs the following:

- For each bit b, if node i has seen a valid $(R+1)$-batch for b, it sets $\mathsf{Extracted}_i \leftarrow \mathsf{Extracted}_i \cup \{b\}$.
- **Output to $\mathcal{Z}$.** If $|\mathsf{Extracted}_i| = 1$, then it outputs the unique $b_i \in \mathsf{Extracted}_i$ to $\mathcal{Z}$; otherwise, outputs the default value 0 to $\mathcal{Z}$.

Figure 2: **Our protocol.** The protocol is described in the $\mathcal{F}_{\text{mine}}$-hybrid world. Section B will explain how to instantiate $\mathcal{F}_{\text{mine}}$ with cryptographic assumptions.

A.3.1 Assumptions and Notations

Additional terminology for corruption status. To make our proofs more precise, we define some additional terminology. At any time in the protocol, nodes that remain honest so far are referred to as *so-far honest* nodes; nodes that remain honest till the end of the protocol are referred to as *forever honest* nodes.

Committee. Without loss of generality, we consider a modification to the protocol, where $\mathcal{F}_{\text{mine}}$ flips a coin for each (b, i) pair upfront. When $\mathcal{F}_{\text{mine}}$ receives `mine` queries, it simply retrieves the corresponding coin that has already been flipped earlier. In this world, we can define the notion of *committees* more easily: For each bit b, a node $i \neq 1$ is in the committee $\mathtt{Com}_b$ for b, if $\mathsf{Coin}[b, i] = 1$.

Honest and corrupt votes. A (valid) vote for a bit b from a node i is said to be honest if the node is so-far honest at the moment the vote is cast, which is the moment when node i calls $\mathcal{F}_{\text{mine}}.\mathtt{mine}(b)$; otherwise, the (valid) vote is said to be corrupt or dishonest.

Handling signature failure. Assume that the signature scheme is secure, and that $\mathcal{A}$ and $\mathcal{Z}$ are probabilistic polynomial time, it must hold that except with negligible probability, no so-far honest node should have a forged signature in view. This is formalized in the following fact:

Fact 3 (No signature failure)**.** *Assume that the signature scheme is secure. Then, except with negligible probability over the choice of* view*, the following holds: at any time in* view*, if the sender is so-far honest and did not sign the bit* $b \in \{0, 1\}$*, then no so-far honest node has seen a valid signature on* b *from the sender.*

Proof. By straightforward reduction to signature security. □

There are two types of bad events that can cause our protocol's security to fail: 1) signature failure (captured by Fact 3); and 2) other stochastic bad events related to $\mathcal{F}_{\text{mine}}$'s coin flips. In the next subsection, we will bound the probability of the latter type of bad events — and there we will pretend that the signature scheme is "ideal" and there are no signature failures — but we stress that technically, we are actually taking a union bound over signature failure and the stochastic bad events analyzed in the next subsection.

A.3.2 Proofs: Bounding Stochastic Bad Events

The protocol clearly satisfies termination; and validity also follows trivially from Fact 3. Thus the remainder of this section will focus on the consistency proof. Our proofs work for general choices of parameters, including the honest fraction ϵ (which can be a function of n) and the failure probability δ. As a special case, assuming that ϵ is any arbitrarily small positive constant and moreover, the mining difficulty parameter p and the total number of stages R are chosen as in Figure 2, then the failure probability $\delta = e^{-\omega(\log \kappa)}$ would be a negligible function in the security parameter κ.

To prove consistency, we will prove that there is no *discrepancy* for either bit (except with δ probability), which is formally defined as follows.

Discrepancy for b. A *discrepancy* for $b \in \{0, 1\}$ occurs if at the end of the protocol there exist two honest nodes such that b is in exactly one of the two corresponding extracted sets. We further classify the following two types of discrepancy if, in addition, the following conditions are satisfied.

Type-A. A type-A discrepancy for b occurs when b is first added to some honest node's extracted set in some stage in $[1..R]$.

Type-B. A type-B discrepancy for value b occurs when $R + 1$ is the only stage in which b is added to any honest node's extracted set.

Fact 4. *If an honest vote is cast for value b (at the second round of some stage) during the protocol, then for each forever honest node i, it holds at termination that $b \in \mathsf{Extracted}_i$.*

Lemma 4 (Type-A Discrepancy)**.** *Suppose the probability of success for $\mathcal{F}_{\text{mine}}.\mathtt{mine}(\cdot)$ is $p := \min\{1, \frac{1}{\epsilon n} \cdot \log \frac{1}{\delta}\}$. For any value b, a type-A discrepancy for value b happens with probability at most δ.*

Proof. It suffices to prove the claim that if a type-A discrepancy for value b occurs, then there are at least ϵn nodes, each of which has called $\mathcal{F}_{\text{mine}}.\mathtt{mine}(b)$ with unsuccessful result at some moment when it is still so-far honest. For the trivial case $e^{-\epsilon n} \geq \delta$, we have $p = 1$ and every mining attempt must be successful. For the case $e^{-\epsilon n} < \delta$, this event happens with probability at most $(1-p)^{\epsilon n} \leq \exp(-\epsilon n p) \leq \delta$, which implies the result of the lemma.

The rest of the proof establishes the above claim. Observe that a type-A discrepancy implies that at some moment, there is a first time when an so-far honest node adds b to its extracted set in some stage $r \in [1..R]$. If this happened in the **second** round of stage r, then this so-far honest node is in $\mathtt{Com}_b$ and would have been able to cast a valid $(r+1)$-batch of votes that can be seen by everyone in stage $r+1$. Therefore, a type-A discrepancy means that b is first added to an so-far honest node i in the **first** round of stage r, which means node i has seen some valid r-batch of votes.

Since this node i is so-far honest, it will multicast this batch to everyone, and every so-far honest node that has not tried to call $\mathcal{F}_{\text{mine}}.\mathtt{mine}(b)$ before will call $\mathcal{F}_{\text{mine}}.\mathtt{mine}(b)$ in the second round of stage r.

Since a type-A discrepancy occurs, it must be case that all (previous or present) trials of $\mathcal{F}_{\text{mine}}.\mathtt{mine}(b)$ by so-far honest nodes have returned unsuccessful. Since at any moment, the number of so-far honest nodes is at least ϵn, we conclude that the claim is true, and this completes the proof. □

Fact 5 (Chernoff Bound)**.** *Suppose X is the sum of independent $\{0,1\}$-independent random variables. Then, for any $\tau > 0$, the following holds:*

$$\Pr[X \geq (1+\tau)E[X]] \leq \exp(-\frac{\tau \cdot \min\{\tau, 1\} \cdot E[X]}{3})$$

Lemma 5 (Type-B Discrepancy)**.** *Let $p := \min\{1, \frac{1}{\epsilon n} \log \frac{1}{\delta}\}$ as in Lemma 4, and set $R := \lceil \frac{3}{\epsilon} \cdot \ln \frac{1}{\delta} \rceil$. Then, for any value b, a type-B discrepancy for b happens with probability at most δ.*

Proof. A type-B discrepancy for b occurs implies that some honest node i sees a valid $(R+1)$-batch of votes, which are all cast by dishonest nodes. This is because if one of the votes was cast by an so-far honest node (in the second round) of some stage $r \in [1..R]$, then everyone would have seen a valid $(r+1)$-batch in stage $r+1$, in which case a discrepancy would not have occurred.

Observe that a dishonest vote is cast only if a node is corrupted before it calls $\mathcal{F}_{\text{mine}}.\mathtt{mine}(b)$ for the first time. There are at most $(1-\epsilon)n$ dishonest node and each of them can call $\mathcal{F}_{\text{mine}}.\mathtt{mine}(b)$ successfully independently with probability p. **Important:** Note that even if nodes are corrupted adaptively (for instance, based on mining results of other values), the success probability of mining value b is still p.

For the trivial case, $\delta \leq e^{-\epsilon n}$, $R \geq 3n$ is not interesting; hence, it suffices to consider $\delta > e^{-\epsilon n}$ and $p < 1$. We next consider two cases.
Case $\epsilon \geq \frac{1}{4}$. Set $\tau := \frac{3\epsilon}{1-\epsilon} \geq 1$. By Chernoff Bound, the probability that there are more than $R = \lceil \frac{3}{\epsilon} \cdot \ln \frac{1}{\delta} \rceil \geq (1+\tau)(1-\epsilon)np$ dishonest votes is at most $\exp(-\frac{\tau(1-\epsilon)np}{3}) = \delta$.
Case $\epsilon < \frac{1}{4}$. Set $\tau = 1$. By Chernoff Bound, the probability that there are more than $R = \lceil \frac{3}{\epsilon} \cdot \ln \frac{1}{\delta} \rceil \geq (1+\tau)(1-\epsilon)np$ dishonest votes is at most $\exp(-\frac{(1-\epsilon)np}{3}) \leq \delta$. □

Corollary 1. *Suppose that with probability* 1*, there are at least* ϵ *fraction of forever honest nodes and let* δ *be the desired failure probability. By setting the mining success probability* $p := \min\{1, \frac{1}{\epsilon n} \log \frac{2}{\delta}\}$ *and* $R := \lceil \frac{3}{\epsilon} \cdot \ln \frac{2}{\delta} \rceil$*, the protocol satisfies consistency with probability at least* $1 - \delta$.

Proof. For the trivial case $\frac{\delta}{2} \leq e^{-\epsilon n}$, the bound $R \geq 3n$ is not interesting. Hence, it suffices to consider $\frac{\delta}{2} > e^{-\epsilon n}$ and $p < 1$.

To use union bound over type-A and type-B discrepancy for both values of b, we set the failure probability to be $\frac{\delta}{2}$ in Lemmas 4 and 5. $\square$

B Removing the Idealized Functionality $\mathcal{F}_{\text{mine}}$

So far, we have assumed the existence of an $\mathcal{F}_{\text{mine}}$ ideal functionality. In this section, we describe how to instantiate the protocols in the real world. Our techniques follow the approach described by Abraham et al. [2]. Although this part is not a contribution of our paper, for completeness, we describe all the building blocks and the approach in a self-contained manner.

B.1 Preliminary: Adaptively Secure Non-Interactive Zero-Knowledge Proofs

We use $f(\kappa) \approx g(\kappa)$ to mean that there exists a negligible function $\nu(\kappa)$ such that $|f(\kappa) - g(\kappa)| < \nu(\kappa)$.

A non-interactive proof system henceforth denoted nizk for an NP language $\mathcal{L}$ consists of the following algorithms.

- $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\kappa, \mathcal{L})$: Takes in a security parameter κ, a description of the language $\mathcal{L}$, and generates a common reference string crs.
- $\pi \leftarrow \mathsf{P}(\mathsf{crs}, \mathsf{stmt}, w)$: Takes in crs, a statement stmt, a witness w such that $(\mathsf{stmt}, w) \in \mathcal{L}$, and produces a proof π.
- $b \leftarrow \mathsf{V}(\mathsf{crs}, \mathsf{stmt}, \pi)$: Takes in a crs, a statement stmt, and a proof π, and outputs 0 (reject) or 1 (accept).

Perfect completeness. A non-interactive proof system is said to be perfectly complete, if an honest prover with a valid witness can always convince an honest verifier. More formally, for any $(\mathsf{stmt}, w) \in \mathcal{L}$, we have that

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Gen}(1^\kappa, \mathcal{L}),\ \pi \leftarrow \mathsf{P}(\mathsf{crs}, \mathsf{stmt}, w) : \mathsf{V}(\mathsf{crs}, \mathsf{stmt}, \pi) = 1\right] = 1$$

Non-erasure computational zero-knowledge. Non-erasure zero-knowledge requires that under a simulated CRS, there is a simulated prover that can produce proofs without needing the witness. Further, upon obtaining a valid witness to a statement a-posteriori, the simulated prover can explain the simulated NIZK with the correct witness.

We say that a proof system $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ satisfies non-erasure computational zero-knowledge iff there exists a probabilistic polynomial time algorithms $(\mathsf{Gen}_0, \mathsf{P}_0, \mathsf{Explain})$ such that

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Gen}(1^\kappa), \mathcal{A}^{\mathsf{Real}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 1\right] \approx \Pr\left[(\mathsf{crs}_0, \tau_0) \leftarrow \mathsf{Gen}_0(1^\kappa), \mathcal{A}^{\mathsf{Ideal}(\mathsf{crs}_0,\tau_0,\cdot,\cdot)}(\mathsf{crs}_0) = 1\right],$$

where $\mathsf{Real}(\mathsf{crs}, \mathsf{stmt}, w)$ runs the honest prover $\mathsf{P}(\mathsf{crs}, \mathsf{stmt}, w)$ with randomness r and obtains the proof π, it then outputs (π, r); $\mathsf{Ideal}(\mathsf{crs}_0, \tau_0, \mathsf{stmt}, w)$ runs the simulated prover $\pi \leftarrow \mathsf{P}_0(\mathsf{crs}_0, \tau_0, \mathsf{stmt}, \rho)$ with randomness ρ and without a witness, and then runs $r \leftarrow \mathsf{Explain}(\mathsf{crs}_0, \tau_0, \mathsf{stmt}, w, \rho)$ and outputs (π, r).

Perfect knowledge extration. We say that a proof system $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ satisfies perfect knowledge extraction, if there exists probabilistic polynomial-time algorithms $(\mathsf{Gen}_1, \mathsf{Extr})$, such that for all (even unbounded) adversary $\mathcal{A}$,

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Gen}(1^\kappa) : \mathcal{A}(\mathsf{crs}) = 1\right] = \Pr\left[(\mathsf{crs}_1, \tau_1) \leftarrow \mathsf{Gen}_1(1^\kappa) : \mathcal{A}(\mathsf{crs}_1) = 1\right],$$

and moreover,

$$\Pr\left[(\mathsf{crs}_1, \tau_1) \leftarrow \mathsf{Gen}_1(1^\kappa); (\mathsf{stmt}, \pi) \leftarrow \mathcal{A}(\mathsf{crs}_1); w \leftarrow \mathsf{Extr}(\mathsf{crs}_1, \tau_1, \mathsf{stmt}, \pi) : \begin{array}{l} \mathsf{V}(\mathsf{crs}_1, \mathsf{stmt}, \pi) = 1 \\ \text{but } (\mathsf{stmt}, w) \notin \mathcal{L} \end{array}\right] = 0$$

B.2 Adaptively Secure Non-Interactive Commitment Scheme

An adaptively secure non-interactive commitment scheme consists of the following algorithms:

- $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\kappa)$: Takes in a security parameter κ, and generates a common reference string crs.
- $C \leftarrow \mathsf{com}(\mathsf{crs}, v, \rho)$: Takes in crs, a value v, and a random string ρ, and outputs a committed value C.
- $b \leftarrow \mathsf{ver}(\mathsf{crs}, C, v, \rho)$: Takes in a crs, a commitment C, a purported opening (v, ρ), and outputs 0 (reject) or 1 (accept).

Computationally hiding under selective opening. We say that a commitment scheme $(\mathsf{Gen}, \mathsf{com}, \mathsf{ver})$ is computationally hiding under selective opening, iff there exists a probabilistic polynomial time algorithms $(\mathsf{Gen}_0, \mathsf{com}_0, \mathsf{Explain})$ such that

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Gen}(1^\kappa), \mathcal{A}^{\mathsf{Real}(\mathsf{crs}, \cdot)}(\mathsf{crs}) = 1\right] \approx \Pr\left[(\mathsf{crs}_0, \tau_0) \leftarrow \mathsf{Gen}_0(1^\kappa), \mathcal{A}^{\mathsf{Ideal}(\mathsf{crs}_0, \tau_0, \cdot)}(\mathsf{crs}_0) = 1\right]$$

where $\mathsf{Real}(\mathsf{crs}, v)$ runs the honest algorithm $\mathsf{com}(\mathsf{crs}, v, r)$ with randomness r and obtains the commitment C, it then outputs (C, r); $\mathsf{Ideal}(\mathsf{crs}_0, \tau_0, v)$ runs the simulated algorithm $C \leftarrow \mathsf{comm}_0(\mathsf{crs}_0, \tau_0, \rho)$ with randomness ρ and without v, and then runs $r \leftarrow \mathsf{Explain}(\mathsf{crs}_0, \tau_0, v, \rho)$ and outputs (C, r).

Perfectly binding. A commitment scheme is said to be perfectly binding iff for every crs in the support of the honest CRS generation algorithm, there does not exist $(v, \rho) \neq (v', \rho')$ such that $\mathsf{com}(\mathsf{crs}, v, \rho) = \mathsf{com}(\mathsf{crs}, v', \rho')$.

Theorem 4 (Instantiation of our NIZK and commitment schemes [10])**.** *Assume standard bilinear group assumptions. Then, there exists a proof system that satisfies perfect completeness, non-erasure computational zero-knowledge, and perfect knowledge extraction. Further, there exist a commitment scheme that is perfectly binding and computationally hiding under selective opening.*

Proof. The existence of such a NIZK scheme was shown by Groth et al. [10] via a building block that they called *homomorphic proof commitment scheme.* This building block can also be used to achieve a commitment scheme with the desired properties. □

B.3 NP Language Used in Our Construction

In our construction, we will use the following NP language $\mathcal{L}$. A pair $(\mathsf{stmt}, w) \in \mathcal{L}$ iff

- parse $\mathsf{stmt} := (\rho, c, \mathsf{crs}_{\mathrm{comm}}, b)$, parse $w := (\mathsf{sk}, s)$;
- it must hold that $c = \mathsf{comm}(\mathsf{crs}_{\mathrm{comm}}, \mathsf{sk}, s)$, and $\mathsf{PRF}_{\mathsf{sk}}(b) = \rho$.

B.4 Removing $\mathcal{F}_{\text{mine}}$ with Cryptography

We can remove the $\mathcal{F}_{\text{mine}}$ oracle by leveraging cryptographic building blocks including a pseudorandom function family, a non-interactive zero-knowledge proof system that satisfies computational zero-knowledge and computational soundness, and a perfectly binding and computationally hiding commitment scheme. Essentially, with these primitives we can construct an appropriate VRF with adaptive security. Note that some earlier works [5, 12] also achieved such an adaptively secure VRF using unique signatures and random oracles. Here we adopt Abraham et al. [2]'s approach since it removes the random oracle assumption.

We now provide a formal description of how to compile our $\mathcal{F}_{\text{mine}}$-hybrid protocols into real-world protocols using cryptography. The intuition is very simple. Every node commits to a PRF secret key in its public key. This committed secret key is used to evaluate a PRF on $b = 0$ or $b = 1$ to determine whether the node belongs to the b-th committee. The node can then prove to everyone that the eligibility determination is performed correctly by employing a NIZK. Below we give a more formal description of how to rely on this idea to compile the earlier $\mathcal{F}_{\text{mine}}$-hybrid protocol to the real world.

- **PKI setup.** Upfront, a trusted party runs the CRS generation algorithms of the commitment and the NIZK scheme to obtain $\mathsf{crs}_{\text{comm}}$ and $\mathsf{crs}_{\text{nizk}}$. It then chooses a secret PRF key for every node, where the i-th node has key sk_i. It publishes $(\mathsf{crs}_{\text{comm}}, \mathsf{crs}_{\text{nizk}})$ as the public parameters, and each node i's public key denoted pk_i is computed as a commitment of sk_i using a random string s_i. The collection of all users' public keys is published to form the PKI, i.e., the mapping from each node i to its public key pk_i is public information. Further, each node i is given the secret key (sk_i, s_i).

- **Instantiating $\mathcal{F}_{\text{mine}}$.`mine`.** Recall that in the ideal-world protocol a node i calls $\mathcal{F}_{\text{mine}}.\mathtt{mine}(b)$ to check if it is in the b-th committee. Now, instead, the node i calls $\rho := \mathsf{PRF}_{\mathsf{sk}_i}(b)$, and computes the NIZK proof
$$\pi := \mathsf{nizk.P}((\rho, \mathsf{pk}_i, \mathsf{crs}_{\text{comm}}, b), (\mathsf{sk}_i, s_i))$$
where s_i the randomness used in committing sk_i during the trusted setup. Intuitively, this zero-knowledge proof proves that the evaluation outcome ρ is correct w.r.t. the node's public key (which is a commitment of its secret key).

  The mining attempt for b is considered successful if $\rho < D_p$ where D_p is an appropriate difficulty parameter such that a random string of appropriate length is less than D_p with probability p — the probability p is selected in the same way as the earlier $\mathcal{F}_{\text{mine}}$-hybrid world in Figure 2.

  Recall that earlier in our $\mathcal{F}_{\text{mine}}$-hybrid protocol, every message multicast by a so-far honest node i is a vote of the form (b, i) where node i has successfully called $\mathcal{F}_{\text{mine}}.\mathtt{mine}(b)$. Each such message (b, i) that node i wants to multicast is translated to the real-world protocol as follows: we rewrite (b, i) as (b, i, ρ, π) where the terms ρ and π are those generated by i in place of calling $\mathcal{F}_{\text{mine}}.\mathtt{mine}(b)$ in the real world (as explained above). Note that in our $\mathcal{F}_{\text{mine}}$-hybrid protocols a node $j \neq i$ may also relay a message (b, i) mined by i — in the real world, node j would be relaying (b, i, ρ, π) instead.

- **Instantiating $\mathcal{F}_{\text{mine}}$.`verify`.** In the $\mathcal{F}_{\text{mine}}$-hybrid world, a node would call $\mathcal{F}_{\text{mine}}.\mathtt{verify}$ to check the validity of votes upon receiving them, In the real-world protocol, we perform the following instead: upon receiving the vote (b, i, ρ, π), a node can verify the vote's validity by checking:

1. $\rho < D_p$ where p is an appropriate difficulty parameter parametrized in the same way as Figure 2 and
2. π is indeed a valid NIZK for the statement formed by the tuple $(\rho, \mathsf{pk}_i, \mathsf{crs}_{\mathrm{comm}}, b)$. The tuple is discarded unless both checks pass.

Now using the same proofs as Abraham et al. [2], we can prove that the compiled real-world protocols enjoy the same security properties as the $\mathcal{F}_{\mathrm{mine}}$-hybrid protocols. Since the proofs follow identically, we omit the details and refer the reader to Abraham et al. [2]. In the following theorem we assume that the pseudorandom function family employed is secure, the non-interactive zero-knowledge proof system employed satisfies computational zero-knowledge and computational soundness, and moreover, the commitment scheme is perfectly binding and computationally hiding.

Theorem 5 (Real-world protocol: restatement of Theorem 2)**.** *Assume that the cryptographic primitives employed are secure in the sense mentioned above. For parameters $\epsilon, \delta \in (0, 1)$ which are allowed to be functions in n, the aforementioned real-world protocol terminates in $O(\log(1/\delta)/\epsilon)$ number of rounds and achieves BA with $1 - \delta - \mathsf{negl}(\kappa)$ probability in the presence of an adversary that adaptively corrupts at most $(1 - \epsilon)n$ nodes and runs in time polynomial in κ.*

Proof. Note our techniques for instantiating $\mathcal{F}_{\mathrm{mine}}$ with actual cryptography is borrowed from Abraham et al. [2]. Their proof for showing that that the real-world protocol preserves the security properties proved in the ideal world is immediately applicable to our case. ☐

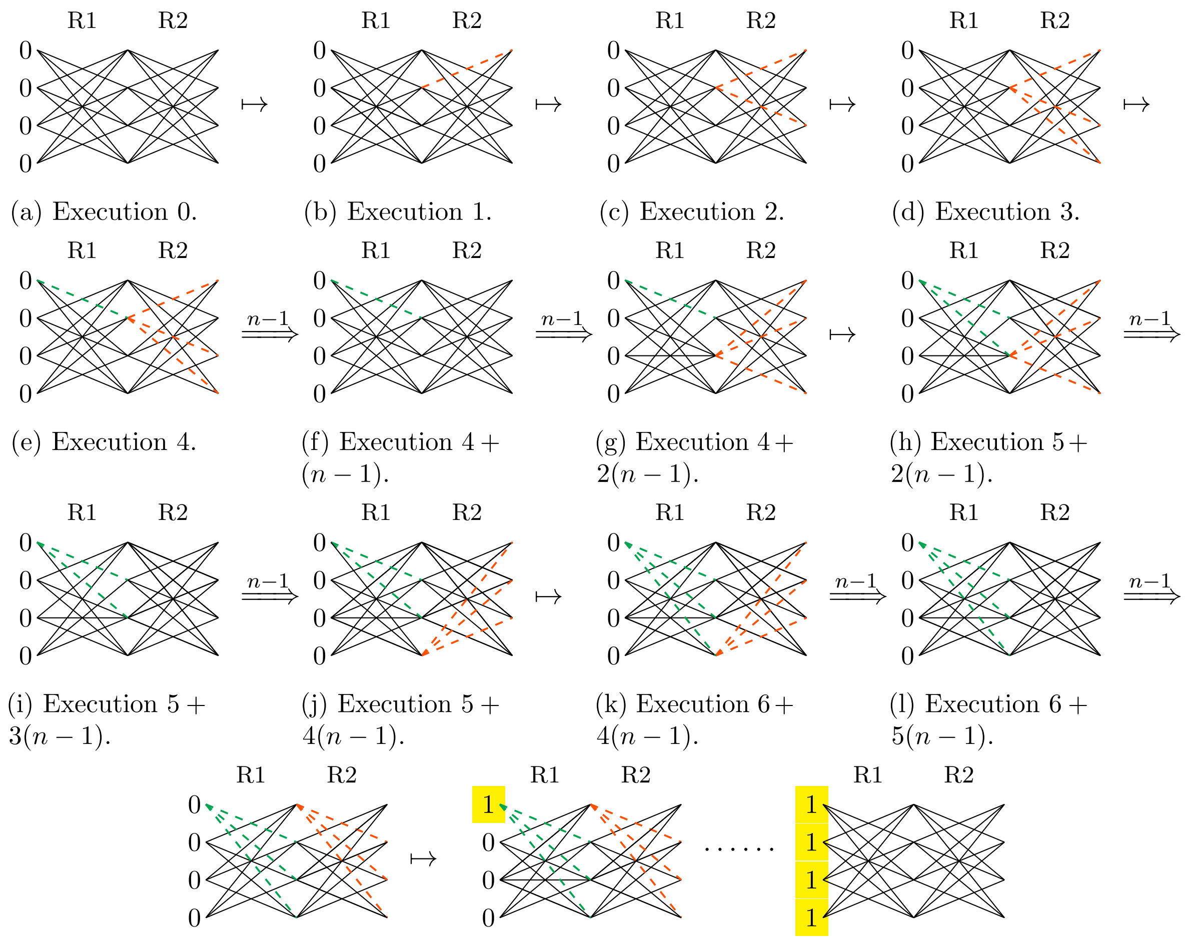
C Background: $f + 1$ Round Complexity Lower Bound for Deterministic Protocols

We begin by reviewing the famous $f + 1$ round complexity lower bound for deterministic protocols. We will present a re-exposition [1] of Dolev and Strong's proof [6] that is conducive to our proof for randomized protocols later. For simplicity, we will first consider the special case for $f = 2$ and later extend the proof to more general f. More concretely, we would like to show that no 2-round deterministic protocol can realize BA in the presence of 2 corruptions. Recall that $n \geq f + 2$. Since we are concerned about the round complexity of the protocol, without loss of generality, we may assume that in every round, everyone sends a message to everyone; for convenience, we assume in this example that nodes do not send messages to themselves.

The proof proceeds in a sequence of executions as depicted in Figure 3. In every execution, every node runs the honest protocol; however, a network adversary may erase some of the messages. The messages erased are denoted by dashed lines in Figure 3.

Alternatively, instead of viewing the attack as being conducted by a network-only adversary, we can imagine that a subset of the nodes are corrupt: corrupt nodes suppress a subset to all of the messages it ought to send but otherwise follow the honest protocol. Thus, if a node i's message is partially or completely being erased in some round (i.e., there exist dashed edges outgoing from i) then node i is treated as corrupt. Henceforth we use the notation $i \xrightarrow{r} j$ to denote the r-th round message from node i to node j where $r \in \{1, 2\}$ and $i, j \in [n]$. Similarly, for $S, S' \subseteq [n]$, $S \xrightarrow{r} S'$ denotes the set of all round-r messages from any node in S to any node in S'. For convenience, we often use $*$ to denote wildcard, i.e., $* := [n]$. We often use $j \in [n]$ to denote a singleton set $\{j\}$ whenever convenient.

Invariants. The sequence of executions in Figure 3 are constructed with two important invariants in mind.



(a) Execution 0. (b) Execution 1. (c) Execution 2. (d) Execution 3.

(e) Execution 4. (f) Execution $4 + (n-1)$. (g) Execution $4 + 2(n-1)$. (h) Execution $5 + 2(n-1)$.

(i) Execution $5 + 3(n-1)$. (j) Execution $5 + 4(n-1)$. (k) Execution $6 + 4(n-1)$. (l) Execution $6 + 5(n-1)$.

(m) Execution $6 + 6(n-1)$. (n) Execution $7 + 6(n-1)$. (o) Final execution.

Figure 3: Sequence of executions for proving the famous $f+1$ lower bound for deterministic protocols. Dashed lines denote communication that is removed by the adversary. The arrow $\mapsto$ denotes a single transition to the next execution in this sequence (also called a "small step"); the arrow $\xRightarrow{n-1}$ denotes a collection of $n-1$ transitions (also called a "big step") where the $n-1$ edges concerned are removed or added back one by one.

1. *Neighboring constraint:* for any pair of adjacent executions, there exists at least one node that is honest in both executions, and moreover its view has not changed in between these executions.

2. *Corruption constraint:* in every execution, there exist at most one sender whose messages can be erased in each round (i.e., only one node can have outgoing dashed edges in each round) — in this way the total number of corrupt nodes is upper bounded by $f = 2$.

Sequence of executions. With these two important invariants in mind, we now describe the sequence shown in Figure 3.

- To start with, in Execution 0, everyone receives the input bit 0 and no message is erased. In this case, everyone should output 0 due to validity.

- In Execution 1, message $2 \xrightarrow{2} 1$ is erased. Now, observe nodes 3 and 4 are honest in both Execution 0 and Execution 1, and moreover, their views have not changed in between the two executions. Since these nodes output 0 in Execution 0, they must output 0 in Execution 1 as well. Since only one node is corrupt in Execution 2, by the consistency requirement, all honest nodes (i.e., everyone except node 2) should output 0 in Execution 1.

- In Executions 2 and 3, we erase the messages $2 \xrightarrow{2} 3$ and $2 \xrightarrow{2} 4$ one by one. In both Executions 2 and 3, only node 2 is corrupt. Moreover, we respect the neighboring constraint as we remove the edges one by one. We thus conclude in a similar fashion that in both Executions 3 and 4, every honest node should output 0.

- In Execution 4, we erase the message $1 \xrightarrow{1} 2$. Since in Execution 3, node 2 does not send any second-round messages, erasing $1 \xrightarrow{1} 2$ does not change node 3 or 4's view in these two adjacent executions. Obviously, Execution 4 respects the corruption constraint. Therefore by a similar argument, we conclude that all honest nodes output 0 in Execution 4.

- Now, we restore the messages $2 \xrightarrow{2} *$ one by one, leading to Execution $4 + (n - 1)$. It is not hard to verify that each step respects both the neighboring and the corruption constraints. Thus all honest nodes must output 0 in these executions.

- Now one by one, we erase the second-round message $3 \xrightarrow{2} *$ leading to Execution $4 + 2(n - 1)$. Both above constraints are respected in each step.

- At this moment, we may erase the message $1 \xrightarrow{1} 3$.

- We now continue this process as shown in Figure 3. When we reach Execution $7 + 6(n - 1)$, node 1 no longer sends any message and thus we can flip its input to 1 without affecting any honest nodes' views.

- We now continue this process to flip every node's input bit in a similar fashion, until we reach the final execution where all nodes are honest and receive the input 1. Since in all steps we respect the neighboring constraint as well as the corruption constraint, we may conclude that even in the final execution all honest nodes must output 0. However, this violates the validity requirement and thus we have reached a contradiction.