Communication-Efficient Byzantine Agreement without Erasures

T-H. Hubert Chan Rafael Pass Elaine Shi

October 16, 2018

Abstract

Byzantine agreement (BA) has always been a classical distributed systems abstraction. About a decade ago, decentralized cryptocurrency and smart contract started to gain popularity, inspiring us to study the agreement problem in large-scale decentralized networks. In such decentralized systems, protocol messages are *multicast* over a peer-to-peer network to all consensus nodes (as opposed to a point-to-point network with pairwise channels). In this paper, we consider BA protocols suitable for such large-scale settings — specifically, protocols that make use of only *a small number of multicast messages* to reach agreement. We say that a BA protocol achieves CC multicast complexity if only CC number of nodes need to speak (i.e., multicast messages) during the entire protocol.

In this paper, we show new feasibility and infeasibility results regarding communication efficient BA protocols in the multicast model. First, we prove that under a plain authenticated channels model without any setup assumptions, BA protocols with sublinear (in the number of nodes) multicast complexity is impossible in the presence of an adaptive adversary, even when 99% of the nodes are honest.

We then show that this infeasibility can be circumvented assuming a public-key infrastructure and standard cryptographic assumptions (and without relying on the ability of honest nodes to securely erase secret data from memory). Specifically, we construct adaptively secure BA protocols with sublinear multicast complexity for either a *synchronous* or a *partially synchronous* network. In a synchronous network, our protocol tolerates up to $\frac{1}{2} - \epsilon$ corruptions (where ϵ is an arbitrarily small constant margin) and in expectation completes in $O(1)$ rounds. In the partially synchronous setting, we construct a communication efficient BA protocol that tolerates up to $\frac{1}{3} - \epsilon$ fraction of corrupt nodes, thus achieving almost optimal resilience.

1 Introduction

Byzantine agreement (BA) [3, 12, 13, 24] is a central abstraction in distributed systems. Roughly speaking, in a BA protocol, every player receives an input bit that is either 0 or 1. The players' goal is to agree on a bit such that the following properties are satisfied over *all but a negligible* (in some security parameter κ) fraction of executions[1]: 1) *consistency:* all honest players should output the same bit b', 2) *validity:* if all honest nodes receive the same input bit b, then all honest nodes must output b.

Today, various cryptocurrency and smart contract platforms repeatedly reach consensus on an Internet scale to maintain an ever-growing, distributed ledger. The communication efficiency of consensus protocols is an important measure of the protocol's scalability in a large-scale deployment.

[1] A Byzantine Agreement problem defined in this manner (called the *agreement* version) is only possible when assuming honest majority. However, there is a variant called Byzantine Broadcast or the *broadcast* version that is possible even under corrupt majority in a synchronous network. In the honest majority setting, given an agreement protocol, one can construct broadcast using only a single instance of agreement plus one additional multicast (from the sender). See Appendix A.2.3 for more discussions.

In existing decentralized cryptocurrency systems, protocol messages are propagated over a peer-to-peer diffusion network to consensus nodes. For example, peers would use some reconciliation protocol to figure out what transactions or blocks other peers are missing, and then propagate the missing data to others. Abstractly, we will model such a communication medium as a *multicast* channel, i.e., when a sender multicasts a message, this message is disseminated to all nodes. While many works in the classical distributed systems literature consider a point-to-point network, in this work, we focus on the communication efficiency of consensus protocols in the *multicast* model. If a consensus protocol requires honest nodes to multicast no more than CC bits of messages, we say that the protocol has *multicast complexity* CC. A BA protocol is communication efficient, if it has sublinear (in the number of nodes) multicast complexity, i.e., if only $o(1)$ fraction of nodes need to speak during the protocol. We further compare the multicast model vs. the classical pairwise channels model in Section 1.4.

In this paper, we explore the multicast complexity of BA protocols by asking the following question:

> *Does there exist a BA protocol with sublinear (in the number of nodes) multicast complexity under standard assumptions?*

We specifically care about answering the above question under an *adaptive* adversary who is capable of corrupting nodes in the middle of protocol execution (however, the adversary has a budget on how many nodes it is allowed to corrupt). In particular, with a *static* adversary who makes corruption choices prior to the start of the protocol, BA with sublinear multicast complexity is relatively easy assuming honest (super-)majority: we can randomly select a small, polylogarithmically-sized committee of players and then run any BA protocol among the committee, and finally simply have the committee members send their outputs to all other non-committee players who could then output the majority decision. Various elegant works have investigated how to achieve random committee election with small communication complexity [8, 15, 17, 19, 21, 22]. Such a committee-based approach, however, fails if we consider an *adaptive attacker*. Such an attacker can simply notice what nodes are on the committee, and then corrupt them, and thereby control the whole committee! Therefore, in the remainder of the paper we focus on the feasibility and infeasibility of BA with sublinear multicast complexity under an *adaptive* adversary.

1.1 Our Results and Contributions

We prove both new upper- and lower-bound results which we summarize below.

Main result 1: impossibility without setup assumptions. First, we show that in the plain *authenticated channels* model without any setup assumptions, it is impossible to achieve adaptively secure BA with sublinear multicast complexity. We state the result in the following theorem — recall that throughout our paper, n denotes the number of consensus nodes and κ denotes the security parameter and our BA protocol retain security except with $1 - \mathsf{negl}(\kappa)$ probability:

Theorem 1. *In a plain authenticated channels model without a PKI, no protocol with $\mathsf{CC}(\kappa, n)$ multicast complexity can achieve BA under $\mathsf{CC}(\kappa, n)$ adaptive corruptions, assuming that the total number of nodes $n = \mathsf{poly}(\kappa)$ is a sufficiently large polynomial. Further, the lower bound holds even assuming fully synchronous communication, the existence of a random oracle or a common reference string, and even in the erasure model.*

Our lower bound proof involves interesting new techniques (particularly, regarding the way we make use of adaptive corruptions) that non-trivially extend classical techniques for proving

consensus lower bounds in the authenticated channels model [16, 23, 24]. We refer the reader to Section 1.3for an informal overview of our lower bound techniques.

In light of the above lower bound, an interesting question is whether with appropriate setup assumptions and standard cryptographic assumptions, BA with sublinear multicast complexity is possible. We show feasibility results for both synchronous and partially synchronous networks. Below we summarize our upper-bound results and compare our results with the prior state-of-the-art.

Main result 2: communication-efficient BA in a synchronous network. Assuming the existence of a public-key infrastructure (PKI) and standard cryptographic assumptions, we demonstrate a synchronous BA protocol with close-to-optimal multicast complexity tolerating $f < n/2$ adaptive corruptions, as summarized in the following theorem.

Theorem 2. *Under standard cryptographic hardness assumptions (more precisely, standard bilinear group assumptions) and assuming the existence of a PKI, for any constant $\epsilon > 0$, there exists a synchronous BA protocol with multicast complexity $\mathsf{poly}\log(\kappa) \cdot \chi$ where n is the number of players, κ is the security parameter, and χ is a cryptographic security parameter[2]; the protocol tolerates $f < (1-\epsilon)n/2$ adaptively corrupted players, and only requires an expected $O(1)$ number of rounds.*

Prior to our work, the closest related results were described in very recent works by Chen and Micali [9], and later extended by several others [10, 20, 25, 26] — these works are the first to demonstrate BA protocols with sublinear multicast complexity; however, *all of them make strong assumptions* including the ability of honest nodes to erase secret data from memory (often called the *erasure model* in the standard cryptography literature [6]), as well as the existence of *Random Oracles (RO)*. Among these works, Chen and Micali consider an adversary that can adaptively corrupt less than $\frac{1}{3}$ fraction of nodes; but subsequent works extend the feasibility results to minority corruptions. Interestingly, just like our work, all of these works [9, 10, 20, 25, 26] were motivated by the scalability of large-scale consensus (e.g., consensus protocols that underly decentralized cryptocurrency systems). In comparison with the related works [9,10,20,25,26], our work removes these strong assumptions including the erasure of secret data from memory and random oracles, and we prove our result under standard cryptographic assumptions.

Main result 3: communication-efficient BA in a partially synchronous network. In a partially synchronous network [13], there is some maximum message delay denoted Δ that is a polynomial function in some security parameter κ; however, the protocol is unaware of Δ.

As far as we know, there is no known partially synchronous, adaptively secure BA protocol with sublinear multicast complexity. Indeed, all known protocols [7, 13] (with adaptive security) require every honest node to speak. We present the first adaptively secure, partially synchronous BA protocol with sublinear multicast complexity, as summarized in the following theorem.

Theorem 3. *Under the same assumptions as Theorem 2, for any constant $\epsilon > 0$, there exists a partially synchronous BA protocol with multicast complexity $\mathsf{poly}\log(\kappa) \cdot \chi$, where n is the number of players, κ is the security parameter, and χ is a cryptographic security parameter; the protocol tolerates $f < (1-\epsilon)n/3$ adaptively corrupted players[3].*

[2] χ would be of $\mathsf{poly}\log(\kappa)$ bits in length if we assumed subexponential security of the cryptographic primitives employed.

[3] Throughout the paper, we assume that a protocol's execution is parametrized with n, κ, χ, and ϵ. We assume that n is a polynomial function in κ.

We remark that in a partially synchronous network, BA is impossible if at least $n/3$ of the nodes are corrupt [13], and this lower bound holds even under static corruption and assuming the existence of PKI. Therefore, we achieve almost optimal resilience.

Terminology. In the remainder of the introduction, we will give an informal overview of our technical results. We use the terminology *forever-honest* to refer to a node that remains honest till it terminates the protocol execution; and we use the terminology "a *so-far-honest* takes some action" to mean that the node has not been corrupt yet when it takes the action (but may become corrupt later), and thus the action is honest.

1.2 Upper Bounds: Key Idea in a Nutshell

We first explain the key ideas behind our upper bound results. Our idea is a general method for taking certain types of BA protocols with high communication complexity, appropriately select a subset of players to participate in the protocol, and then appropriately transmit the result of the protocol to everyone else. As we already mentioned, for the case of static security, this is not hard to do (at least with the appropriate trusted set-up, and assuming an honest majority). Chen and Micali [9] were the first to show how to apply this subset-selection idea and achieve adaptive security, but assuming erasures and RO. We now need to achieve adaptive security but *without erasures*. Towards doing this, we will restrict our attention to certain voting-based protocols: roughly speaking, in each round, the players have some "preferred" value b in mind which they "vote" for. In between rounds (based on nodes' view in the protocol so far) they may change the bit. This process continues until some event happens (e.g., some pre-defined termination round is reached, or some "good event" happens), and the players eventually output their final bit b.

Given such a voting-based protocol, the key idea for reducing the communication complexity is to restrict, by random subset-selection, the set of players that are *eligible to vote* at each round. The crux is how to determine eligibility.

Strawman: the Chen-Micali approach. A first approach (similar to Chen and Micali's approach [9]) would be to determine eligibility by applying a verifiable random function (VRF) to the round number and the player identity, and determining that a player is eligible if the output is smaller than some value D. The VRF requires knowing the player's secret key to be evaluated. Thus only the player itself knows at what rounds it is eligible to vote; but anyone knowing only the player's public key can check that the the evaluation was correctly done when given the output of the VRF.

The problem with this is that once an adaptive attacker notices that some player i was eligible to vote in round r (because i sent a legal vote), the attacker can corrupt i immediately and make i vote for a different bit in the same round! To tackle this issue, Chen and Micali [9] requires that the player employ a *forward-secure* signing scheme[4], and *erase* its round-specific secret-key immediately after casting the vote, such that even if the attacker instantly corrupts this node, it cannot cast a vote for a different b in the same round.

Our key insight: vote-specific eligibility. Our approach is different and does not rely on erasures. The key idea is to employ a *vote-specific eligibility determination*: whether you are

[4]Informally, in a forward secure signing scheme, in the beginning the node has a key that can sign any slot numbered 0 or higher; after signing a message for slot t, the node can update its key to one that can henceforth sign only slots $t+1$ or higher, and the old key is erased.

eligible to vote in round r is a function not only of r and your identity, but also of the bit b you want to vote for. More precisely, we now apply the player's VRF[5] to the pair (r, b).

What does this achieve? Suppose that the attacker sees somebody, say, player i, votes for the bit b in round r. Although the attacker can corrupt i immediately after it speaks in round r, the only thing this can definitely guarantee is to make player i vote again for the same bit b in round r, but the honest player has already done that! Moreover, the fact that i was allowed to vote for b in round r, does not make i any more likely to be eligible to vote for $1 - b$ in round r (or vote for either bit in any future round). Thus, corrupting player i is basically equivalent to corrupting a random player.

Technical challenges. Obviously, this overview significantly oversimplifies. First, we need to come up with appropriate voting-based protocols for the above-mentioned settings. Secondly, although the above intuition can be formalized in a "idealized" model of "secret subset-selection"—we indeed define an idealized "secret mining" functionality which captures our usages of the VRF—it turns out to be non-trivial to formally prove the security of this approach. The problem is related to the famous "selective-opening" problem [14]—it becomes non-trivial to ensure the security of players' VRF instances when an attacker can selectively ask to see the secret keys for VRFs of its choice. Indeed, a similar problem arises in Chen and Micali's work [9] and he resorted to the use of a Random Oracle. We instead present a method for overcoming this in the standard model by combining a PRF (pseudorandom function), and adaptively secure, non-interactive commitments and zero knowledge proofs.

1.3 Lower Bound for Authenticated Channels: Informal Overview

For the plain authenticated channels model without any setup assumptions, we show the impossibility of sublinear multicast-complexity BA. Our proof is inspired by the classical techniques for proving consensus lower bounds in the authenticated channels model [16,23,24]; however, we extend known techniques in novel and non-trivial manners, particularly in the way we rely on the ability to make adaptive corruptions to complete the proof. We provide an informal overview of our lower bound proof below (for Theorem 1).

For our lower bound, we consider a *broadcast* variant of the BA problem (which as explained below, makes our lower bound stronger). In the broadcast version of the problem, there is a *designated sender* who tries to send its input bit to all players. *Consistency* requires that (except with negligible probability) all honest nodes output the same bit; and *validity* requires that (except with negligible probability) if the sender is forever-honest, all honest nodes must output the sender's input bit. In a synchronous network with honest majority, one can construct broadcast using a single instance of agreement plus one additional multicast message from the sender — see Appendix A.2.3 for more discussions. Thus proving a lower bound for the broadcast version (for synchrony and any constant fraction of corruption) makes our lower bound stronger.

Suppose that some protocol achieves adaptive security and sublinear multicast complexity. We describe a hypothetical experiment: consider two honest executions that share a single node (that is not the designated sender): `(input: 0) Q --- 1 --- Q' (input: 1)`. The set Q contains nodes numbered $2, \ldots, n$, and so does the set Q'. The node 1 is shared across the two executions. Whenever a node in Q (or Q' resp.) sends a message, all nodes in Q (or Q' resp.) and the node 1 receives the message. Whenever 1 wants to send a message, it sends it to nodes in both Q and Q'. If 1 receives a message from either $i \in Q$ or $i \in Q'$, it acts as if the message is received from i. We

[5]This is an informal characterization of our technique. Later for different protocols, this VRF is actually applied to different terms but the bit b being voted on is always part of the eligibility election.

assume that $2 \in Q$ and $2 \in Q'$ are the senders respectively in the two executions, and that they receive the inputs 0 and 1 respectively.

We now interpret this hypothetical experiment in two ways. First, it could be that 1 is the malicious node simulating all of Q' and Q is honest. Alternatively, it could be that there are only nodes 1 to n, initially all honest, and the nodes in Q' are imaginary and entirely simulated by the adversary in its head. Whenever some node in Q' wants to speak, the corresponding one in Q is adaptively corrupt to implement this action — it is not difficult to see that the adversary needs to corrupt only sublinear number of nodes. By the validity requirement in the former interpretation (where 1 is corrupt), we conclude that nodes in Q must output 0 and nodes in Q' must output 1 by symmetry. Now, consider the latter interpretation (where 1 is honest), we may conclude that the node 1 must be consistent with nodes in Q; and by symmetry 1 must be consistent with nodes in Q' too. This allows us to reach a contradiction and rule out the existence of such a protocol.

1.4 Relations and Implications for the Classical Pairwise Channels Model

Note that if we had a classical, pairwise-channels model where each pair of nodes can transmit messages over a pairwise link, we could implement a multicast from a single honest sender as $\Theta(n)$ messages from the sender to every recipient. Thus a protocol with CC multicast complexity directly implies a protocol with $\mathsf{CC} \cdot O(n)$ *pairwise communication complexity*. On the other hand, in general a protocol with $\mathsf{CC} \cdot n$ pairwise communication complexity does not imply a protocol with CC multicast complexity.

Our upper bounds lead to new results in the classical, pairwise-channels model too. In this model, Dolev and Reischuk [11] proved a quadratic lower bound for the pairwise communication complexity of any *deterministic* BA protocol. For a long time, an open question was whether, allowing the use of randomness, BA can be achieved with sub-quadratic pairwise communication complexity. A partial answer to this question was provided in a breakthrough result by King and Saia [21] in 2010. They presented a randomized synchronous BA protocol with pairwise communication complexity $O(n^{1.5})$ while tolerating $\frac{1}{3}$ fraction of adaptive corruptions. Their protocol, however, must also rely on the *erasure model*. In comparison, our upper bound results imply BA protocols with $\tilde{O}(n)$ pairwise communication complexity[6]: *i)* for synchronous networks, tolerating minority adaptive corruptions; and *ii)* for partially synchrony, tolerating $\frac{1}{3}$ adaptive corruptions.

Thus in the synchronous setting we not only improve the communication complexity and resilience of King and Saia [21], but also remove the erasure assumption; on the other hand we introduce standard cryptographic assumptions[7] while King and Saia's result does not make complexity assumptions (but assumes private channels). For the partially synchronous setting, to the best of our knowledge no prior result was known that achieves subquadratic pairwise communication complexity (and simultaneously adaptive security under standard assumptions) — and thus our result is the first of this nature.

2 Impossibility in the Authenticated Channels Model

In this section, we show that it is impossible to have an adaptively secure Byzantine agreement protocol that achieves $o(n)$ multicast complexity under an authenticated channels model (i.e., without PKI) — even when 99% of the nodes must remain honest. Our lower bound holds even when assuming the existence of an RO and in the erasure model.

[6]The notation $\tilde{O}(\cdot)$ hides polylogarithmic factors.

[7]In comparison, the recent works [9, 10, 20, 26] improve the pairwise communication complexity and/or resilience over King and Saia [21], but all these works still assume erasure and moreover they require a Random Oracle (RO).

As mentioned in Section 1.3, for our lower bound, we consider a *broadcast* variant of the BA problem. In the broadcast version of the problem, there is a designated sender who tries to send its input bit to all players. *Consistency* requires that (except with negligible probability) all honest nodes output the same bit; and *validity* requires that (except with negligible probability) if the sender is forever-honest, all honest nodes must output the sender's input bit. As we explained earlier in Section 1.3, considering the broadcast variant makes our lower bound stronger.

Model for our lower bound. We consider a model where any message multicast by an honest sender is delivered to all honest nodes at the beginning of the next round, i.e., $\Delta = 1$. Further, the message always carries the true identity of the sender, i.e., the communication channel authenticates the sender.

Proof of Theorem 1. We now prove the impossibility result, i.e., Theorem 1 whose statement was formally presented in the introduction.

Suppose for the sake of contradiction that there exists a Byzantine broadcast protocol with $\mathsf{CC}(\kappa, n)$ multicast complexity and tolerating $\mathsf{CC}(\kappa, n)$ corruptions; and suppose that $n = \mathsf{poly}(\kappa)$ is a sufficiently large polynomial. We will prove the impossibility by considering a hypothetical experiment where a set of $2n - 1$ nodes, connected in specific ways, execute the honest protocol. We argue that this hypothetical experiment can be interpreted in two different ways; and relying on the security properties of BA, we reach a contradiction by reasoning about the two interpretations.

A hypothetical experiment. Consider a hypothetical experiment depicted in the following graph:

```
(input: 0) Q --- 1 --- Q' (input: 1)
```

More specifically, imagine that the node 1 simultaneously participates in two executions of the protocol: on the left the node 1 plays with the set Q, containing nodes numbered $2, 3, \ldots, n$; on the right, the node 1 plays with the set Q', containing nodes also numbered $2, 3, \ldots, n$. Suppose that node $2 \in Q$ and node $2 \in Q'$ are the two senders; further $2 \in Q$ receives the input 0 and $2 \in Q'$ receives the input 1. All nodes in $Q \cup \{1\} \cup Q'$ execute the *honest* protocol where messages are forwarded as below:

- whenever node 1 multicasts a message, the same message is delivered to nodes in both Q and Q';
- whenever any node $i \in \{2, \ldots, n\}$ from Q multicasts a message, it is delivered to all nodes in Q as well as the node 1;
- whenever any node $i \in \{2, \ldots, n\}$ from Q' multicasts a message, it is delivered to all nodes in Q' as well as the node 1.

Note that node 1 treats a message from $i \in Q$ and $i \in Q'$ identically, i.e., the message has the sender identity i for some $i \in \{2, \ldots, n\}$ no matter whether i comes from Q or Q'.

To summarize the above defines a hypothetical experiment containing $2n-1$ nodes all executing the honest protocol and where protocol messages are routed in specific ways. We now interpret this hypothetical experiment in two different manners, leading to a contradiction.

Corrupt-1 interpretation: First, the hypothetical experiment can be viewed as an execution among n nodes numbered $1, 2, \ldots, n$, where the node 1 is (statically) corrupt. Specifically, imagine that the set Q' represents the honest nodes numbered $2, 3, \ldots, n$; and imagine that the corrupt node 1 is simulating all nodes in Q in its head. In this case, by the definition of multicast complexity, we immediately conclude the following where we use the random variable view to denote a random sample of the hypothetical experiment:

Claim 1. *Except with negligible probability over the choice of* view *of the hypothetical experiment, nodes in* Q' *cannot send more than* $\mathsf{CC}(\kappa, n)$ *bits of distinct messages. By symmetry, we also have that except with negligible probability over the choice of* view *of the merged execution, nodes in* Q *cannot send more than* $\mathsf{CC}(\kappa, n)$ *bits of distinct messages.*

Now, by the validity requirement of Byzantine broadcast, we have the following:

Claim 2. *Except with negligible probability over the choice of* view *of the hypothetical experiment, all nodes in* Q' *output* 1*. By symmetry, we also have that except with negligible probability over the choice of* view *of the merged execution, all nodes in* Q *output* 0*.*

Now, it is interesting to consider what node 1 should output in the hypothetical experiment. To answer this question, we can view the execution in a different light. In comparison with the earlier interpretation, here the important difference is that now we want to explain the execution assuming 1 is actually honest.

Honest-1 interpretation: Only node 1 and nodes in Q are real and initially honest, and Q' is imaginary and simulated by the adversary in its head — initially, at protocol start, the adversary has not corrupted any node yet. It will make corruption actions along the way.

Note that the adversary can observe all messages sent over the authenticated channel. Based on the messages received, it simulates the actions of the imaginary nodes in Q' assuming that Q' follows the honest protocol. Whenever some node $j \in Q'$ wants to speak, the adversary adaptively corrupts the corresponding node $j \in Q$ and implements this action.

Specifically, at the end of each round, the adversary simulates the next round of all nodes in Q' in its head, and checks to see which nodes are about to send a message in the next round. The adversary then corrupts precisely those nodes that are about to send a message (unless they are already corrupt). As a special case, for the first round of the protocol, the adversary simulates the actions of Q' in its head prior to protocol start and corrupts those that are about to send a message in the first round.

Now, say that at the beginning of some round r, some subset $S \subseteq Q'$ are about to send a message, and recall that by the attack defined above, S has been corrupt by the adversary in Q. At this moment, the corrupt nodes S splits themselves into two threads: one thread still follows the honest protocol and sends whatever message the honest protocol instructs them to send; and the other thread sends whatever message its simulated copy in Q' ought to send in this round — but only to node 1 and not to anyone else.

Since the protocol has $\mathsf{CC}(\kappa, n)$ multicast complexity, except with negligible probability over the choice of view, the adversary will corrupt no more than $\mathsf{CC}(\kappa, n)$ nodes. Due to Claim 2, the remaining honest nodes in Q would output 0 except with negligible probability over the choice of view. By the consistency requirement of Byzantine broadcast[8], except with negligible probability over the choice of view, node 1 should agree with the remaining honest nodes in Q, i.e., output 0 too.

[8]The sender may have been corrupt by the adversary, thus our argument does not rely on validity here.

By symmetry, it could be that the adversary is actually simulating Q and corrupting the corresponding nodes in Q' when nodes in Q want to speak. Due to Claim 2 and the consistency requirement, we conclude that except with negligible probability over the choice of view, node 1 should output 1 (to be consistent with the honest nodes in Q'. Thus we have a contradiction.

3 Communication-Efficient BA for Synchronous Networks

In this section, we present our *synchronous* BA protocol that achieves sublinear multicast complexity. For ease of exposition, in the main body we opt for simplicity: we explain a simple protocol that tolerates only $\frac{1}{3} - \epsilon$ fraction of adaptive corruptions. We note that this simple protocol (that suffers in resilience) advances the theoretical state-of-the-art: to the best of our knowledge no protocol can achieve sublinear multicast complexity in the presence of any constant fraction of adaptive corruptions.

We defer to Appendix B to present a stronger result for synchrony, that is, a sublinear-multicast-complexity BA protocol that tolerates $\frac{1}{2} - \epsilon$ fraction of adaptive corruptions. For ease of exposition, in this section we focus on an intuitive description of the protocol and proof; a *formal* statement of our results and proofs are also deferred to the appendices.

Roadmap of this section. We first describe an extremely simple, communication-inefficient synchronous BA protocol (inspired by the phase king paradigm [2]) that tolerates less than $\frac{1}{3}$ corruptions. Next, we show how to apply our vote-specific eligibility election idea to obtain communication efficiency.

3.1 Preliminary: A Simple, Communication-Inefficient BA Tolerating $\frac{1}{3}$ Corruptions

We provide a very simple "voting-based" BA that is secure as long as $n > 3f$; later we shall see how to improve its multicast complexity using vote-specific eligibility election.

The protocol proceeds in epochs $r = 0, 1, \ldots R-1$ where R is a super-logarithmic function in κ, and every epoch consists of $O(1)$ synchronous rounds. In each epoch r, a particular node is selected to be the "leader": for concreteness, in epoch r, we let node r be the leader. At initialization, every node i sets b_i to its input bit, and sets its "sticky flag" $F = 1$ (think of the sticky flag as indicating whether to "stick" to the bit in the previous epoch).

Each epoch r now proceeds as follows where all messages are signed, and only messages with valid signatures are accepted:

1. The leader of epoch r (i.e., node r) flips a random coin b and multicasts $(\texttt{propose}, r, b)$.
2. Every node i sets $b_i^* := b_i$ if $F = 1$ or if it has not heard a valid proposal from the current epoch's leader[9]. Else, it sets $b_i^* := b$ where b is the proposal heard from the current epoch's leader (if proposals for both $b = 0$ and $b = 1$ have been observed, choose an arbitrary bit).

   Then, node i multicasts $(\texttt{ACK}, r, b_i^*)$.
3. If at least $\frac{2n}{3}$ number of ACKs from distinct nodes have been received and vouch for the same b^*, set $b_i := b^*$ and $F := 1$; else set $F := 0$.

At the end of $R = \omega(\log \kappa)$ epochs, each node outputs the bit that it last sent an ACK for (and 0 if if never sent an ACK message).

[9] We stress that since honest nodes' sticky bit is set to 1 initially, in the first epoch every honest node votes on its input bit. This matters to the validity of the protocol.

Basically, in every epoch, every node either switches to the leader's proposal (if any has been observed) or it sticks to its previous "belief" b_i. This simple protocol works because of the following observations. Henceforth, we refer to a collection of at least $\frac{2n}{3}$ number of ACKs from distinct nodes for the same epoch and the same b as *ample ACKs for* b.

- *Consistency within an epoch.* Suppose that in epoch r, honest node i observes ample ACKs for b from a set of nodes denoted S, and honest node j observes ample ACKs for b' from the set S'. By a standard quorum intersection argument, $S \cap S'$ must contain at least one forever-honest node. Since honest nodes vote uniquely, it must be that $b = b'$.
- *A good epoch exists.* Next, suppose that in some epoch r the leader is honest. We say that this leader chooses a lucky bit b^* iff either 1) in epoch $r-1$, no honest nodes have seen ample ACKs for either bit and thus all honest nodes will switch to the leader's proposal in epoch r; or 2) in epoch $r-1$, some honest nodes have seen ample ACKs for a unique bit b^* (which agrees with the current leader's random choice). Clearly, an honest leader chooses a lucky b^* with probability at least 1/2; and except with $\exp(\Omega(-R))$ probability, an honest-leader epoch with a lucky choice must exist.
- *Persistence of honest choice after a good epoch.* Now, as soon as we reach an epoch (denoted r) with an honest leader and its choice of bit b^* is lucky, then all honest nodes will ACK b^* in epoch r. Thus all honest nodes will hear ample ACKs for b^* in epoch r; therefore, they will all stick to ACKing b^* in epoch $r+1$. By induction, in all future epochs they will stick to ACKing b^*.
- *Validity.* So far we have argued consistency. Validity is also easy: if all honest nodes receive the same bit b^* as input then due to the same argument as above the bit b^* will always stick around in all epochs.

3.2 Communication Efficiency through Vote-Specific Eligibility

The above simple protocol requires at least linear multicast communication (in each round everyone multicasts a message). We now use the vote-specific eligibility determination to elect who is eligible for *proposing* 0 and 1 respectively in each epoch, as well as who is eligible for *ACKing* 0 and 1 respectively in each epoch.

Realizing vote-specific eligibility election. To achieve such vote-specific eligibility determination, we assume a trusted setup: loosely speaking, each node i obtains a random secret sk_i, and a commitment of sk_i will be published in a public-key infrastructure (PKI). We now say that node i is eligible to propose or ACK the bit $b \in \{0,1\}$ in epoch r iff

$$\mathsf{PRF}_{\mathsf{sk}_i}(\texttt{propose}, r, b) < D_0 \quad \text{or} \quad \mathsf{PRF}_{\mathsf{sk}_i}(\texttt{ACK}, r, b) < D_1 \quad \text{resp.}$$

where D_0 and D_1 (called the *difficulty* parameters) are some appropriately chosen parameters, and PRF is a pseudorandom function.

Note that no one else can compute whether i is eligible to propose/ACK b in some epoch r, since only i knows its secret sk_i. For node i to convince others that it indeed is eligible, it can however provide a non-interactive zero-knowledge (NIZK) proof asserting that the above relation holds w.r.t. the secret sk_i whose commitment is contained in the PKI.

Terminology. Henceforth, we call an attempt for node i to check eligibility to send either a `propose` or `ACK` message a *mining* attempt for a `propose` or `ACK` message — this is inspired by Bitcoin's terminology where miners "mine" blocks.

Remark 1 (Subtleties in reasoning about cryptographic schemes)**.** *In the appendices, we will show that to achieve adaptive security, the NIZK and the commitment schemes employed must also provide an adaptive notion of security. Such NIZK and commitment schemes have been shown to exist assuming standard cryptographic assumptions and the existence of a common reference string (CRS). This CRS also needs to be included in the public parameters (besides nodes' public keys) in our formal scheme description later.*

Difficulty parameters. It remains to explain how we set the difficulty parameters:

1. In expectation a single node is elected to send a `propose` message every two epochs (in an honest execution), i.e., each `propose` mining attempt succeeds with probability $\frac{1}{2n}$.
2. In expectation $\lambda = \omega(\log \kappa)$ nodes (i.e., super-logarithmic in κ number of nodes) are elected to send an `ACK` message in every epoch (in an honest execution), i.e., each `ACK` mining attempt succeeds with probability $\frac{\lambda}{n}$.

Communication-efficient BA protocol. We use the phrase "node i *conditionally multicasts* a message $(\mathtt{T}, r, b)$" to mean that node i checks if it is eligible to vote for b in epoch r and if so, it multicasts $(\mathtt{T}, r, b, i, \pi)$, where π is a NIZK proof proving that i indeed is eligible. Here $\mathtt{T} \in \{\mathtt{propose}, \mathtt{ACK}\}$ stands for the type of the message.

Now, our new subset-sampling based protocol is almost identical to simple protocol with large multicast complexity except for the following changes:

- every occurrence of multicast is now replaced with "conditionally multicast";
- the threshold number of ACKs for a bit to stick is now replaced with $\frac{2\lambda}{3}$; and
- upon receiving every message, a node checks the NIZK to verify the message's validity (whereas in the earlier protocol nodes only checks signatures).

Intuitive analysis. We now explain why our new protocol works, by following similar arguments as the underlying BA — but now we must additionally analyze the stochastic process induced by eligibility election.

To help our analysis, we shall abstract away the cryptography needed for eligibility election, and instead think of eligibility election as making "mining" queries with a trusted party called $\mathcal{F}_{\text{mine}}$. Specifically, if a node i wants to check its eligibility for $(\mathtt{T}, r, b)$ where $\mathtt{T} \in \{\mathtt{propose}, \mathtt{ACK}\}$, it calls $\mathcal{F}_{\text{mine}}.\mathtt{mine}(\mathtt{T}, r, b)$, and $\mathcal{F}_{\text{mine}}$ shall flip a random coin with appropriate probability to determine whether this "mining" attempt is successful. If successful, $\mathcal{F}_{\text{mine}}.\mathtt{verify}((\mathtt{T}, r, b), i)$ can vouch to any node of the successful attempt (imagine that this is used in place of verifying the NIZK proof). We now analyze this stochastic process.

- *Consistency within an epoch.* We first argue why "consistency within an epoch" still holds with the new scheme. Henceforth, if a node makes a mining attempt for some $(\mathtt{T}, r, b)$ while still being honest, this is called an *honest mining attempt* (even if the node immediately becomes corrupt afterwards in the same round). Else, if an already corrupt node makes a mining attempt, it is called a *corrupt mining attempt*.

  There are at most $(\frac{1}{3} - \epsilon)n$ corrupt nodes, each of which might try to mine for 2 ACKs (one for each bit) in some fixed epoch r. On the other hand, each so-far-honest node will try to mine for only 1 ACK in each epoch. Therefore, in epoch r, the total number of (honest or corrupt) mining attempts is at most $(\frac{1}{3} - \epsilon)n \cdot 2 + (\frac{2}{3} + \epsilon)n \cdot 1 = (\frac{4}{3} - \epsilon)n$, each of which is **independently** successful with probability $\frac{\lambda}{n}$.

Hence, if there are $\frac{2\lambda}{3}$ ACKs for each of the bits 0 and 1, this means there are at least in total $\frac{4\lambda}{3}$ successful mining attempts, which happens with negligible probability, by the Chernoff Bound. Therefore, except with negligible probability, if any node sees $\frac{2\lambda}{3}$ ACKs for some bit b, then no other node sees $\frac{2\lambda}{3}$ ACKs for a different bit b'.

Remark 2. *It is important that the eligibility election be tied to the bit being proposed/ACKed. Had it not been the case, the adversary could observe whenever an honest node sends* $(\texttt{ACK}, r, b)$*, and immediately corrupt the node in the same round and make it send* $(\texttt{ACK}, r, 1-b)$ *too. In this case, clearly if there are* $\frac{2\lambda}{3}$ *ACKs for* b *in epoch* r*, then by corrupting all these nodes that sent the ACKs, the adversary can construct* $\frac{2\lambda}{3}$ *ACKs for* $1-b$*, and thus "consistency within an epoch" does not hold.*

- *A good epoch exists.* We now argue why "a good epoch exists" in our new scheme. Here, for an epoch r to be good, the following must hold: 1) a single so-far-honest node successfully mines a `propose` message, and no already corrupt node successfully mines a `propose` message[10]; and 2) if some honest nodes want to stick to a (unique) belief b^* in epoch r, the leader's random coin must agree with b^*. Note that every so-far-honest node makes only one `propose` mining attempt per epoch. Every already corrupt node can make two `propose` mining attempts in an epoch, one for each bit. Regardless, recall that our `propose` mining difficulty parameter is set such that on average one node is elected leader every 2 epochs (in an honest execution) — this implies that in every epoch, with $\Theta(1)$ probability, a single honest `propose` mining attempt is successful and no corrupt `propose` mining attempt is successful. Since our protocol consists of $\lambda = \omega(\log \kappa)$ epochs, a good epoch exists except with negligible in κ probability.
- *Persistence of honest choice after a good epoch and validity.* Finally, the remainder of the proof, including "persistence of honest choice after a good epoch" and "validity" hold in a relatively straightforward fashion by applying the standard Chernoff bound.

4 Communication-Efficient BA for Partial Synchrony

We describe a partially synchronous BA protocol that tolerates $\frac{1}{3} - \epsilon$ adaptive corruptions and achieves sublinear multicast complexity. To the best of our knowledge, this is the first sublinear multicast complexity protocol in the partially synchronous model.

4.1 Warmup: Communication-Inefficient Underlying BA

To illustrate the intuition, we first describe a simple communication-inefficient BA protocol that works for partial synchrony. We think that this protocol is folklore (e.g., the highest-epoch POP idea was also used by Abraham et al. [1] but they consider a state machine replication abstraction rather than one-shot consensus), but we have not seen an explicit description: we believe that specifying this simple protocol makes a contribution too; in particular, it can be used for pedagogical purpose. This simple BA protocol is inspired by the simple synchronous BA protocol in Section 3.

Additional simplifying assumption. We make a couple simplifying assumptions here (which can easily be removed later). We will for the time being allow ourselves to neglect the validity requirement and focus on how to achieve consistency. Further, we will allow the protocol to run

[10] If in some epoch multiple honest nodes mine a `propose` message, we can just think of this epoch as having a corrupt proposer.

forever, but nodes are guaranteed to produce an output in polynomial time as long as $n \geq 3f + 1$, where n denotes the total number of nodes and f denotes the number of adaptively corrupt nodes. Finally, we will assume an ideal random leader election oracle: at the beginning of each epoch (to be defined later), the oracle randomly chooses a node to be the leader, and all nodes are notified of this decision at the beginning of the epoch.

Definition of epochs and leader. Let λ be a super-logarithmic function in the security parameter κ. Imagine that time progresses in $n\lambda$ epochs of length 1, followed by $n\lambda$ epochs of length 2, $n\lambda$ epochs of length 4, and so on. We assume that an ideal random leader election oracle as described above. Henceforth, the leader of epoch r is said to be the r-th leader.

Definition of POP. Any collection of $2f + 1$ epoch-r prepare messages from distinct nodes is said to be an epoch-r proof-of-preparation (POP).

The underlying BA. The protocol works as follows — we stress that if during epoch r, a node receives propose and prepare message for epoch $r' \neq r$, such messages get discarded without being processed.

- *Propose.* When epoch r starts, if the r-th leader has seen a POP, let E be the POP with the highest epoch number and let b be the bit vouched for by E: the r-th leader now multicasts $(\texttt{propose}, r, b, E)$. Else, if it has not seen any valid POP, it picks a random bit $b \in \{0, 1\}$, and multicasts $(\texttt{propose}, r, b, \perp)$.
- *Prepare.* When the first proposal $(\texttt{propose}, r, b, \_)$ is heard from the correct leader *during* epoch r, vote as follows (and all future proposals heard are ignored): if the node has not seen a valid POP from past epochs, it multicasts a signed vote $(\texttt{prepare}, r, b)$; else it multicasts a signed vote $(\texttt{prepare}, r, b)$ where b is bit vouched for by the POP of the highest epoch seen so far (including the one possibly attached as evidence in the present proposal).
- *Report.* If *during* epoch r, a node hears $2f + 1$ epoch-r prepare messages from distinct nodes vouching for the same bit b (i.e., an epoch-r POP for b), it multicasts a signed report $(\texttt{report}, r, b)$.
- *Finalize.* If a node *ever* (i.e., not necessarily in epoch r) hears $2f + 1$ epoch-r reports from distinct nodes vouching for the same bit b (and for any but the same r), it outputs b but continues participating in the protocol.

Intuitive analysis. We now argue why the above protocol achieves consistency and why honest nodes will output a bit in polynomial time (although the protocol runs forever). For simplicity, we assume ideal signature in the following analysis.

- *Consistency within an epoch.* In any epoch r, any two valid POPs must vouch for the same bit. This arises from a standard quorum intersection argument, i.e., any two sets of $2f + 1$ so-far-honest nodes must intersect at one (or more) so-far-honest node(s) — and this node in the intersection votes uniquely in any epoch.
- *Consistency across epochs.* Now, observe that if any so-far-honest node outputs a bit b, then it must be that it has seen $2f + 1$ report messages from distinct nodes for some epoch r, and $f + 1$ of them must be sent by forever-honest nodes, and by the end of epoch r. This means that $f + 1$ forever-honest nodes must have seen an epoch-r POP for b. Now, in epoch $r + 1$, these $f + 1$ forever-honest nodes will not send prepare messages for $1 - b$, and thus epoch $r + 1$ cannot gain a POP for $1 - b$. By induction we can conclude that no epoch $r' \geq r$ can gain a POP for $1 - b$.

Now, we argue why honest nodes will output a bit in polynomial time. Since the epoch length doubles every n epochs, in polynomial number of epochs, the epoch length will be larger than 10Δ where Δ is the maximum network delay that is unknown to the protocol. Consider the time when the epoch length first becomes at least 10Δ, and let r be one of the $n\lambda$ epochs whose duration first becomes at least 10Δ. At the beginning of this epoch r, either 1) all forever-honest nodes have not seen any valid POP; or 2) some have seen a valid POP. In the former case, if the epoch's leader is forever-honest, then every forever-honest node will send a prepare message for the honest leader's proposed bit b and within the epoch, all forever-honest nodes will hear $2f+1$ reports on b. Now in $n\lambda$ epochs there must exist an epoch with a forever-honest leader except with $\exp(-\Omega(\lambda))$ probability. In the latter case, recall that there will be $n\lambda$ epochs of the same length. In every such epoch, with probability at least $1/n$, an epoch will be *lucky*, i.e., a forever-honest node who has seen the highest-epoch POP among all forever-honest nodes will be elected leader. If this happens, in this lucky epoch, every forever-honest node will send a prepare message for the leader's proposed bit b; and within the same epoch every forever-honest node will have seen $2f+1$ reports on b. Now in $n\lambda$ epochs of length 10Δ, except with $\exp(-\Omega(\lambda))$ probability, there must be such a lucky epoch. Thus, honest nodes are guaranteed to output a bit in polynomial time except with $\exp(-\Omega(\lambda))$ probability.

4.2 Communication Efficiency with Vote-Specific Eligibility Election

We will now describe how to employ vote-specific eligibility election, how to set difficulty parameters, and how to adjust epoch lengths to achieve communication efficiency.

Eligibility election. In the new protocol, nodes need to mine a ticket (i.e., determine its eligibility) before sending any message. Nodes perform eligibility election for three types of messages $(\texttt{propose}, r, b)$, $(\texttt{prepare}, r, b)$, and $(\texttt{report}, r, b)$ — and this is achieved by relying on a PRF, a commitment and a NIZK scheme just like in Section 3.2. Note how the eligibility election is tied to the epoch r and the bit b the node would like to vouch for. Thus it remains to specify how to set the difficulty parameters: we will set parameters such that every `propose` message is successfully mined with probability $p_0 := \frac{1}{2n}$, and every `prepare` or `report` message is successfully mined with probability $p := \frac{\lambda}{n}$ where $\lambda = \omega(\log \kappa)$ is any super-logarithmic function in the security parameter κ.

New threshold parameter for POP. In light of the random eligibility election, we redefine the POP parameter too: for any epoch r, and either bit b, a collection of at least $\mathcal{T} = 2np/3$ valid `prepare` messages from distinct *eligible* nodes form an epoch-r POP for b — recall that p is the probability that each `prepare` mining attempt is successful.

Epochs. We will now redefine epoch lengths such that the same length repeats fewer number of times before doubling. Specifically, there will be $R = \Theta(\lambda)$ epochs of length 1, followed by R epochs of length 2, followed by R epochs of length 4, and so on.

Achieving validity with initial reports. Our underlying simple BA scheme earlier did not achieve validity. We will now fix this problem by introducing *initial reports*. Upfront in the protocol, each node, if eligible, signs its input bit and announces it at the beginning of the protocol. Initial reports can be considered as epoch-0 reports, and they are mined with the same probability as any other report. The idea is that before proposers see any POPs, they can include a sufficiently

large collection of initial reports in a proposal as evidence. Henceforth, E is valid evidence for a proposal $(\texttt{propose}, r, b)$ iff

- Either E is a collection of at least $\mathcal{T}$ initial reports from distinct *eligible* nodes, among which a strict majority vouch for the proposed bit b — without loss of generality, we can assume that p is set such that $\mathcal{T}$ is odd, and thus majority must exist; or
- E is a valid epoch-r POP for the bit b (see above for the definition of a valid POP).

Putting it altogether. We are now ready to describe the protocol by assembling the above ideas. Before doing so, we first clarify a few terms and assumptions. We use the term "conditionally multicast $(\texttt{T}, r, b)$" where $\texttt{T} \in \{\texttt{propose}, \texttt{prepare}, \texttt{report}\}$ to denote the fact that a node first determines its eligibility for the message $(\texttt{T}, r, b)$, and if eligible, multicasts the message as well as a zero-knowledge proof (ZKP) of eligibility. Further, any received message is only valid if the ZKP verifies and invalid messages are ignored (i.e., dropped without being processed); additionally, any valid `propose` must also carry a valid proposal evidence. We now present the protocol:

- *Initial report.* At the beginning of the protocol, every node conditionally multicasts $(\texttt{report}, 0, b)$ where b denotes its input bit.
- *Propose.* Whenever each epoch r starts, every node first finds a bit b that has a valid proposal evidence E as follows: if the node has seen a POP from a past epoch, pick E to be the POP of the highest such epoch, and let b be the bit E vouches for; else let E be any set of $\mathcal{T}$ valid initial reports from distinct eligible nodes, and let b be the bit it vouches for. If such a pair (b, E) is found, conditionally multicast $(\texttt{propose}, r, b, E)$.
- *Prepare.* When the first valid proposal $(\texttt{propose}, r, b, \_)$ is heard, vote as follows (and all future proposals heard are ignored): if a node has not seen a valid POP from past epochs, conditionally multicast a vote $(\texttt{prepare}, r, b)$; else conditionally multicast $(\texttt{prepare}, r, b)$ where b is bit vouched for by the POP of the highest epoch seen so far (including the one possibly attached as evidence in the present proposal).
- *Report.* If *during* epoch r, a node collects a valid epoch-r POP for some bit b, it conditionally multicasts the report $(\texttt{report}, r, b)$.
- *Finalize.* If a node *ever* (i.e., not necessarily in epoch r) hears $\mathcal{T}$ valid epoch-r reports from distinct eligible nodes vouching for the same bit b (and for any but the same r), it outputs the bit b but continue participating (we shall discuss how to terminate later).

Informal stochastic analysis. Consistency can be argued in a similar fashion as for the simplified version of the protocol, but the quorum-intersection type arguments will now need to be of a probabilistic nature. For example, suppose that a so-far-honest node i has observed $\frac{2np}{3}$ number of epoch-r prepare messages for b, we can conclude that at least $(\frac{1}{3} + 0.9\epsilon) \cdot n$ number of so-far-honest nodes (denoted the set S) must have attempted to mine a ticket for $(\texttt{prepare}, r, b)$. Similarly, if another so-far-honest node j has observed $\mathcal{T}$ number of epoch-r prepare messages for b', then at least $(\frac{1+\epsilon}{3} + 0.9\epsilon) \cdot n$ so-far-honest nodes (denoted the set S') must have attempted to mine a ticket for $(\texttt{prepare}, r, b')$. Now by the pigeon-hole principle, $S \cap S'$ must be non-empty, and we can conclude that $b = b'$ by observing that any so-far-honest node will make a mining attempt for $(\texttt{prepare}, r, b)$ for only a single b. Of course, the node can get corrupted immediately upon sending a $(\texttt{prepare}, r, b)$ message and be made to mine for $(\texttt{prepare}, r, 1-b)$ — here importantly, since our committee election is *bit-specific*, the chance that this node successfully mines a ticket for $(\texttt{prepare}, r, 1-b)$ is no better than any other node. This explains again why bit-specificity

is important in the eligibility determination. We defer the formal consistency proof to the next subsection.

To argue liveness, we need to argue that a good epoch exists in which *a)* a single so-far-honest proposer is elected and no already corrupt node is elected a proposer; *b)* the epoch length is sufficiently long; and *c)* either no honest node one has seen any POP by the beginning of the epoch, or the honest proposer has a POP whose epoch number is at least as large as $2n/3$ forever-honest nodes. Recall that had all nodes been honest, on average, one proposer is successfully elected every 2 epochs. This means that except with negligible in κ probability, every $R = \omega(\frac{\log \kappa}{\epsilon})$ epochs, there must exist one in which a single so-far-honest proposer and no corrupt proposer is elected; and conditioned on this there is at least $\Theta(\epsilon)$ probability that condition (c) is satisfied. Now, since the epoch length doubles every R epochs, after a while, the epoch length will become sufficiently long.

Early termination. So far in the protocol, nodes participate forever even after having output a bit. It is not difficult to use a standard early termination technique so nodes can terminate immediately upon outputting, that is, upon outputting b, the node conditionally multicasts a *universal* message $(\texttt{universal}, r, b, E)$ after determining its eligibility for $(\texttt{universal}, r, b)$ — here E is a set of $\mathcal{T}$ epoch-r reports that cause the node to output. The probability of mining a universal message is the same as `prepare` or `report`. Such a valid universal message can in any future epoch $r' > r$ serve as a replacement for any message from this node, including `propose`, `prepare`, and `report`.

We stress that although this early termination technique is standard in the consensus literature, it turns out that when combining with our eligibility election technique, the stochastic analysis becomes somewhat subtle — now, all the `propose`, `prepare`, and `report` that the universal message substitutes share randomness. In the appendices, we argue that such dependence does not matter to our stochastic reasoning. The intuition is that our proofs only require that for a fixed message category T and epoch r, the mining events over different nodes i and values b are independent. On the contrary, we do not need independence of mining events over different message categories and epochs. Finally, we note that if we use the universal message to substitute `propose` messages, it would seem like this has changed the probability of mining `propose` — but since consistency is guaranteed for valid universal messages, this increase in probability for mining `propose` does not matter. With the above protocol and early termination technique, we can prove Theorem 3 (presented in the introduction) — we defer the proofs to the appendices.

Summary of Deferred Materials in Appendices

In our main body, we described our results slightly informally to aid understanding. To formalize these results require significantly more efforts — in particular, the cryptographic primitives employed for realizing vote-specific eligibility determination must be adaptively secure; and our computational reduction proofs under an adaptive adversary are subtle.

We give an overview of the additional results and formal proofs contained in the appendices.

1. We show how to improve the resilience of our synchronous protocol to tolerate minority corruptions and formal proofs in Appendix B.

2. We defer a formal description of our partially synchronous protocol and its proofs to Appendix C.

3. Finally, also for ease of exposition, in our appendices we first describe our protocols assuming an $\mathcal{F}_{\text{mine}}$ ideal functionality. We formally describe how to remove this ideal oracle using

appropriate, adaptively secure cryptographic building blocks in Appendix D, and present the computational reduction proofs in Appendix E.

References

[1] I. Abraham, G. Gueta, and D. Malkhi. Hot-stuff the linear, optimal-resilience, one-message bft devil. https://arxiv.org/pdf/1803.05069.pdf.

[2] H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons, Inc., USA, 2004.

[3] M. Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *PODC*, 1983.

[4] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 2000.

[5] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, 2001.

[6] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *STOC*, 1996.

[7] M. Castro and B. Liskov. Practical byzantine fault tolerance. In *OSDI*, 1999.

[8] N. Chandran, W. Chongchitmate, J. A. Garay, S. Goldwasser, R. Ostrovsky, and V. Zikas. The hidden graph model: Communication locality and optimal resiliency with adaptive faults. In *ITCS*, 2015.

[9] J. Chen and S. Micali. Algorand: The efficient and democratic ledger. https://arxiv.org/abs/1607.01341, 2016.

[10] B. M. David, P. Gazi, A. Kiayias, and A. Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Eurocrypt*, 2018.

[11] D. Dolev and R. Reischuk. Bounds on information exchange for byzantine agreement. *J. ACM*, 32(1):191–204, Jan. 1985.

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

[13] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 1988.

[14] C. Dwork, M. Naor, O. Reingold, and L. J. Stockmeyer. Magic functions. *J. ACM*, 2003.

[15] U. Feige. Noncryptographic selection protocols. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, pages 142–, Washington, DC, USA, 1999. IEEE Computer Society.

[16] M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. In *PODC*, 1985.

[17] S. Gilbert and D. R. Kowalski. Distributed agreement with optimal communication complexity. 2010.

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

[19] B. M. Kapron, D. Kempe, V. King, J. Saia, and V. Sanwalani. Fast asynchronous byzantine agreement and leader election with full information. *ACM Trans. Algorithms*, 6(4):68:1–68:28, Sept. 2010.

[20] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Crypto*, 2017.

[21] V. King and J. Saia. Breaking the $O(N^2)$ bit barrier: Scalable byzantine agreement with an adaptive adversary. *J. ACM*, 58(4):18:1–18:24, July 2011.

[22] V. King, J. Saia, V. Sanwalani, and E. Vee. Scalable leader election. In *SODA*, 2006.

[23] L. Lamport. The weak byzantine generals problem. *J. ACM*, 30(3):668–676, 1983.

[24] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[25] S. Micali and V. Vaikuntanathan. Optimal and player-replaceable consensus with an honest majority. MIT CSAIL Technical Report, 2017-004, 2017.

[26] R. Pass and E. Shi. The sleepy model of consensus. In *Asiacrypt*, 2017.

[27] L. Ren, K. Nayak, I. Abraham, and S. Devadas. Practical synchronous byzantine consensus. Cryptology ePrint Archive, Report 2017/307, 2017.

A Preliminaries

A.1 Model of Protocol Execution

We assume a standard protocol execution model with n parties (also called *nodes*) numbered $0, 1, \ldots, n-1$. 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 $\mathcal{A}$ can *adaptively* corrupt nodes any time during the execution. All nodes that have been corrupt 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. Henceforth, 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; nodes that become corrupt before the end of the protocol are referred to as *eventually-corrupt* nodes. Henceforth, we assume that all parties as well as $\mathcal{A}$ and $\mathcal{Z}$ are Interactive Turing Machines, and the execution is parametrized by a security parameter κ that is common knowledge to all parties as well as $\mathcal{A}$ and $\mathcal{Z}$.

Communication model. We assume that the execution proceeds in *rounds*. We say that the network is *synchronous*, if every message sent by an so-far-honest node is guaranteed to be received by an honest recipient at the beginning of the next round. We say that the network is *partially synchronous*, if there is some polynomial function $\Delta(\kappa, n)$ such that every message sent by an so-far-honest node will be delivered to an honest recipient in $\Delta(\kappa, n)$ rounds — however, the protocol is unaware of the choice of $\Delta(\kappa, n)$. In other words, a partially synchronous protocol is supposed to guarantee security for any fixed polynomial function $\Delta(\kappa, n)$.

All of our protocols will be in the *multicast* model: honest nodes participate in the protocol by multicasting messages to each other. We assume that when a so-far-honest node i multicasts a message M, it can immediately become corrupt in the same round and made to send one or more messages in the same round. However, the message M that was already multicast before i became corrupt *cannot be retracted* — in a synchronous model, all other so-far-honest nodes will receive the message M at the beginning of the next round; and in a partially synchronous model, in at most Δ rounds.

Notational conventions. Since all parties, including the adversary $\mathcal{A}$ and the environment $\mathcal{Z}$ are assumed to be non-uniform probabilitic polynomial-time (p.p.t.) Interactive Turing Machines (ITMs), protocol execution is assumed to be probabilistic in nature. We would like to ensure that certain security properties such as consistency and liveness hold for almost all execution traces, assuming that both $\mathcal{A}$ and $\mathcal{Z}$ are polynomially bounded.

Henceforth in the paper, we use the notation $\mathsf{view} \leftarrow \mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$ to denote a sample of the randomized execution of the protocol Π with $\mathcal{A}$ and $\mathcal{Z}$, and security parameter $\kappa \in \mathbb{N}$. The randomness in the experiment comes from honest nodes' randomness, $\mathcal{A}$, and $\mathcal{Z}$, and view is sometimes also referred to as an execution trace or a sample path. We would like that the fraction of sample paths that fail to satisfy relevant security properties be negligibly small in the security parameter κ.

More formally, let P be a polynomial-time computable predicate defined over a view that checks whether certain security properties hold for a view. Whenever we say "except for a *negligible* fraction of the views, $P(\mathsf{view}) = 1$" or "except with *negligible* probability over the choice of view, $P(\mathsf{view}) = 1$", we technically mean the following:

> For any p.p.t. $(\mathcal{A}, \mathcal{Z})$ (possibly required to respect a certain corruption budget), there exists a negligible function $\mathsf{negl}(\cdot)$, such that
>
> $$\Pr\left[\mathsf{view} \leftarrow \mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa) : P(\mathsf{view}) = 1\right] \geq 1 - \mathsf{negl}(\kappa)$$

In particular, a function $\mathsf{negl}(\cdot)$ is said to be negligible if for every polynomial $p(\cdot)$, there exists some κ_0 such that $\mathsf{negl}(\kappa) \leq \frac{1}{p(\kappa)}$ for every $\kappa \geq \kappa_0$.

Definition 1 ((n, α)-respecting)**.** *We say that $(\mathcal{A}, \mathcal{Z})$ is (n, α)-respecting with respect to some protocol Π* iff *for every view in the support of $\mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$, $(\mathcal{A}, \mathcal{Z})$ spawns n nodes among which at most α fraction can be adaptively corrupt.*

Multicast complexity. We now formally define the notion of *multicast complexity*.

Definition 2 (Multicast complexity)**.** *Suppose that Π is a protocol in the multicast model. We say that Π has multicast complexity $\mathsf{CC}(\kappa, n)$ w.r.t. $(\mathcal{A}, \mathcal{Z})$, iff except with negligible probability over the choice of view sampled from $\mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$, the total number of bits multicast by honest nodes is bounded by $\mathsf{CC}(\kappa, n)$.*

A.2 Formal Definitions for Byzantine Agreement

We formally define two versions of the problem, a *broadcast* version where only a designated sender has an input; and an *agreement* version where all nodes have input.

A.2.1 Broadcast Version

In a Byzantine *broadcast* protocol, there is a *designated sender* (or simply *sender*) that is part of the common knowledge. We use the convention that node 0 is the 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. A Byzantine broadcast protocol Π must satisfy consistency, validity, and T_{end}-termination. Specifically, let $T_{\text{end}} := \mathsf{poly}(\kappa, n)$ be a polynomial in κ and n, we say that the protocol Π satisfies consistency, validity, and T_{end}-termination with respect to $(\mathcal{A}, \mathcal{Z})$ *iff* there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every κ, except with $\mathsf{negl}(\kappa)$ probability over the choice of $\mathsf{view} \leftarrow_{\$} \mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$, the following properties hold:

- *Consistency.* If a forever-honest node outputs b_i and another forever-honest node outputs b_j to $\mathcal{Z}$, then it must hold that $b_i = b_j$.
- *Validity.* If the sender is forever-honest and the sender's input is b from $\mathcal{Z}$, then all forever-honest nodes must output b to $\mathcal{Z}$.
- T_{end}*-termination.* By the end of round $T_{\text{end}}(\kappa, n)$, all forever-honest nodes output a bit.

We say that a Byzantine broadcast Π satisfies consistency, validity, and T_{end}-termination in (n, α)-environments, iff for every p.p.t. $(\mathcal{A}, \mathcal{Z})$ that is (n, α)-respecting with respect to Π, Π satisfies consistency, validity, or T_{end}-termination with respect to $(\mathcal{A}, \mathcal{Z})$.

A.2.2 Agreement Version

An *agreement* protocol[11] does not have a designated sender. Intead, every honest node receives an input bit from the environment $\mathcal{Z}$. Validity is required only if all honest nodes receive the same input bit b — in this case, honest nodes' ouput is required to match this bit. We provide formal definitions below.

Syntax. Prior to protocol start, every node i receives an input $b_i \in \{0,1\}$ 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. An agreement protocol Π must satisfy consistency, validity, and T_{end}-termination. Specifically, let $T_{\text{end}} := \mathsf{poly}(\kappa, n)$ be a polynomial in κ and n, we say that the protocol Π satisfies consistency, validity, and T_{end}-termination with respect to $(\mathcal{A}, \mathcal{Z})$ *iff* there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every κ, except with $\mathsf{negl}(\kappa)$ probability over the choice of $\mathsf{view} \leftarrow_{\$} \mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$, the following properties hold:

[11] Agreement is also commonly referred to as "consensus" in the distributed systems literature.

- *Consistency.* If a forever-honest node outputs b_i and another forever-honest node outputs b_j to $\mathcal{Z}$, then it must hold that $b_i = b_j$.
- *Validity.* If all forever-honest nodes receive the same input bit b from $\mathcal{Z}$, then all forever-honest nodes must output b to $\mathcal{Z}$.
- T_{end}-*termination.* By the end of round $T_{\text{end}}(\kappa, n)$, all forever-honest nodes output a bit.

We say that an agreement protocol Π satisfies consistency, validity, and T_{end}-termination in (n, α)-environments, iff for every p.p.t. $(\mathcal{A}, \mathcal{Z})$ that is (n, α)-respecting with respect to Π, Π satisfies consistency, validity, or T_{end}-termination with respect to $(\mathcal{A}, \mathcal{Z})$.

A.2.3 Broadcast vs Agreement

In a partially synchronous network, it is well-known that the broadcast version of the definition is impossible: the intuition (which can easily be turned into a formal proof) is that an honest sender with indefinite delay is indistinguishable from a crashed sender — in the former case, roughly speaking, to satisfy validity, nodes must wait for the sender to send its bit; but in the latter the nodes should make progress without being blocked. Therefore we use the agreement version for our partially synchronous results.

In a synchronous network under honest majority[12], one can construct broadcast using a single instance of agreement with an additional multicast from the sender: we can simply have the sender send its input bit to everyone in the first round. In the second round, everyone invokes an agreement protocol, using the bit received from the sender as the input bit (and if nothing is received or both bits are received, then use 0 as the input bit). Thus, in our lower bound (which apply to synchrony under any constant fraction of corruption), we use the broadcast notion — and this makes the lower bound stronger. For our upper bounds, we use the agreement version.

Finally, we note that under honest majority, one can also construct agreement from broadcast but the most straightforward construction is not communication preserving: fork n instances of broadcast where each node acts as sender respectively, and then output the majority bit.

A.3 Ideal Mining Functionality $\mathcal{F}_{\text{mine}}$

Earlier in Section 3.2, we described how to leverage cryptographic building blocks such as PRFs and NIZKs to realize committee/leader election (or eligibility election). For all our protocols, it would be convenient to describe them assuming such eligibility election is a blackbox primitive. We thus introduce an ideal functionality called $\mathcal{F}_{\text{mine}}$ which, informally speaking, captures the cryptographic procedures of random eligibility selection. One can imagine that $\mathcal{F}_{\text{mine}}$ is a trusted party such that whenever a node attempts to mine a ticket for a message type m, $\mathcal{F}_{\text{mine}}$ flips a random coin with an appropriate probability to decide if this mining attempt is successful. $\mathcal{F}_{\text{mine}}$ stores the results of all previous coin flips, such that if a node performs another mining attempt for the same m later, the same result will be used.

Henceforth in our paper, we will first describe all of our protocols in an ideal world assuming the existence of such a trusted party $\mathcal{F}_{\text{mine}}$ (also referred to as $\mathcal{F}_{\text{mine}}$-hybrid protocols in the cryptography literature [4, 5]). Later in Appendix D, we will show that using the cryptographic techniques described in Section 3.2, all of our $\mathcal{F}_{\text{mine}}$-hybrid protocols can be instantiated in a real world where $\mathcal{F}_{\text{mine}}$ does not exist.

[12] Note that the agreement version of the problem is only possible assuming honest majority.

$\mathcal{F}_{\text{mine}}(1^\kappa, \mathcal{P})$

The function $\mathcal{P} : \{0,1\}^* \to [0,1]$ maps each message to some success probability.

- On receive $\texttt{mine}(\mathsf{m})$ from node i for the first time: let $\mathsf{Coin}[\mathsf{m}, i] := \mathsf{Bernoulli}(\mathcal{P}(\mathsf{m}))$ and return $\mathsf{Coin}[\mathsf{m}, i]$.
- On receive $\texttt{verify}(\mathsf{m}, i)$: if $\texttt{mine}(\mathsf{m})$ has been called by node i, return $\mathsf{Coin}[\mathsf{m}, i]$; else return 0.

Figure 1: The mining ideal functionality $\mathcal{F}_{\text{mine}}$.

$\mathcal{F}_{\text{mine}}$ ideal functionality. As shown in Figure 1, the $\mathcal{F}_{\text{mine}}$ ideal functionality has two activation points:

- Whenever a node i calls $\texttt{mine}(\mathsf{m})$ for the first time, $\mathcal{F}_{\text{mine}}$ flips a random coin to decide if node i has successfully mined a ticket for m.
- If node i has called $\texttt{mine}(\mathsf{m})$ and the attempt is successful, anyone can then call $\texttt{verify}(\mathsf{m}, i)$ to ascertain that indeed i has mined a ticket for m.

This $\mathcal{F}_{\text{mine}}$ functionality is secret since if an so-far-honest node i has not attempted to mine a ticket for m, then no corrupt node can learn whether i is in the committee corresponding to m.

B Synchronous Network with Honest Majority

In this section, we describe an adaptively secure Byzantine agreement protocol that is secure in a synchronous network assuming that $\frac{1}{2} + \epsilon$ fraction of the nodes are honest. In comparison with earlier works on Byzantine agreement [9, 20, 25, 26], our protocol does not assume erasures.

For ease of exposition, we first present a *Byzantine broadcast* protocol (rather than the agreement version).

We will first present the detailed protocol in Appendix B.1 — for simplicity the protocol will be presented assuming that an ideal $\mathcal{F}_{\text{mine}}$ functionality exists and later in Appendix D we will be concerned about actually instantiating $\mathcal{F}_{\text{mine}}$ with cryptography. After describing the full protocol, we then explain how to intuitively understand the protocol and argue its security in Appendix B.2. Finally, in Appendix B.3, we will present formal proofs in the $\mathcal{F}_{\text{mine}}$-hybrid world.

B.1 Detailed Protocol

Mining and message validity. The protocol proceeds in *epochs*, each of which consists of several rounds called `propose`, `prepare`, `commit`, and `report` respectively. In the r-th epoch, messages of the form $((\mathtt{T}, r, b, i), E)$ can be sent, where $\mathtt{T}$ is some *category* in $\{\texttt{prepare}, \texttt{commit}, \texttt{report}\}$, $b \in \{0,1\}$, i denotes the purported sender of the message, and E denotes some evidence that will be used to verify message validity.

Upon receiving an epoch-r message of the form $((\mathtt{T}, r, b, i), E)$, a node checks the message's validity by verifying the following:

1. $\mathcal{F}_{\text{mine}}.\texttt{verify}((\mathtt{T}, r, b), i) = 1$; and
2. E is a valid evidence for $(\mathtt{T}, r, b)$ — what it means for E to be a valid evidence for `propose`, `prepare`, and `commit` messages will be defined later in Figure 2.

Any message that is invalid is ignored and discarded immediately.

Conditionally multicast. Let $((\mathtt{T}, r, b, i), E)$ denote a message of epoch r, purported to be sent from i, and possibly attached with some payload E (which can be the relevant evidence). Since in our protocol, a node always calls $\mathcal{F}_{\text{mine}}$ to mine a ticket first before multicasting any message, henceforth we use the term "node i conditionally multicasts a message $((\mathtt{T}, r, b, i), E)$" to mean the following actions:

$$\text{call } \mathcal{F}_{\text{mine}}.\mathtt{mine}(\mathtt{T}, r, b) \text{ and if successful, multicast } ((\mathtt{T}, r, b, i), E).$$

In other words, a committee is characterized by a message category $\mathtt{T}$, epoch number r and bit b, and it contains all nodes that have successful results from calling $\mathcal{F}_{\text{mine}}.\mathtt{mine}(\mathtt{T}, r, b)$.

Detailed protocol and multicast complexity. We provide the formal protocol description in Figure 2. The protocol's parameters are chosen to ensure negligible in κ security failure assuming that the number of corrupt nodes is upper bounded by $(\frac{1}{2} - \epsilon)n$ for an arbitrarily small positive constant ϵ. It is not difficult to see that this protocol requires that so far honest nodes multicast at most polylogarithmically (in κ) many messages. Further, each message's size must be upper bounded by $\mathsf{poly}\log(\kappa) + \chi$ where χ is a security parameter related to the security of the signature scheme. In the following subsection, we will explain intuitively how to intuitively understand and interpret our protocol.

B.2 Intuition

We now intuitively explain why the protocol described in Figure 2 works. To do this, we first explain a communication-inefficient version of the protocol without random eligibility election; we then explain how the random eligibility election helps.

B.2.1 Warmup

As a warmup, we first explain a communication-inefficient variant of the protocol in Figure 2 but removing random eligibility election. This simplified variant is inspired by (but differs in various details from) the works by Micali and Vaikutanathan [25] and by Ren et al. [27] — these modifications are needed to make their schemes [25, 27] compatible with our small-bandwidth techniques and meanwhile achieve expected constant round at the same time.

Specifically, we consider the following simplifications to the protocol in Figure 2:

1. We run the protocol for $R := \lceil \frac{n+1}{2} \rceil \geq f + 1$ epochs, where f is the number of (eventually) corrupt nodes.
2. Imagine that in each epoch r, node r is the only eligible proposer. Hence, in at least one epoch, the proposer is forever-honest.
3. For prepare, commit, and report messages, pretend that there is no more eligibility election: every node will send prepare, commit, and report messages, i.e., every "conditionally multicast" is replaced with "multicast".
4. A node always signs a message before multicasting it.
5. The threshold is set to $\mathcal{T} := f + 1$. We assume $n \geq 2f + 1$, where n is the total number of nodes and f is the number of corrupt nodes.

In this simplified case, to argue that the protocol achieves consistency, a key observation is the following which holds for all but a negligible fraction of view's (due to failure of the signature scheme):

Byzantine Broadcast: Synchronous Network with Honest Majority

Parameters: Let λ be a super-logarithmic function in the security parameter κ. The adversary can adaptively corrupt at most $(\frac{1}{2} - \epsilon)$ fraction of nodes. A propose message is successfully mined with $\frac{1}{2n}$ probability; a prepare, commit, or report message is successfully mined with $p := \frac{\lambda}{\epsilon^2 n}$ probability; the threshold is set to $\mathcal{T} := \frac{np}{2}$; and the number of epochs is set to be $R = \lambda$.

Epoch $r = 1, \ldots, R$**.** Each epoch r consists of exactly 4 synchronous rounds, referred to as `propose`, `prepare`, `commit`, and `report` respectively.

- **Propose**.
  - For $r = 1$, the sender 0 signs its input bit b_0 to produce a signature $\mathtt{Sig}_0(b_0)$; it then multicasts $((\mathtt{propose}, 1, b_0), 0, \mathtt{Sig}_0(b_0))$
  - For $r > 1$, every node i forms an evidence E containing $\mathcal{T}$ number of epoch-r' reports from distinct nodes (including itself) for every $r' < r$. It then chooses one bit b such that E is a valid proposal evidence for $(\mathtt{propose}, r, b)$, and conditionally multicasts $((\mathtt{propose}, r, b, i), E)$. The definition of "valid proposal evidence" is defined as follows.
  - Evidence E is said to be a valid proposal evidence for $(\mathtt{proposal}, r, b)$, iff the following holds:
    i) either $r = 1$ and E is a valid signature from node 0 on b; or
    ii) $r > 1$, E consists of at least $\mathcal{T}$ number of epoch-r' reports for every $r' < r$, and moreover if at least one of the report messages vouches for a non-$\perp$ bit, then b must be one of the non-$\perp$ bits with the maximum epoch number r'. Note that this means that if all report messages in E vouch for $\perp$, then both $(\mathtt{proposal}, r, 0)$ and $(\mathtt{proposal}, r, 1)$ are consistent with E and can be proposed.
- **Prepare.** When a valid propose message of the form $((\mathtt{propose}, r, b, j), \_)$ for some $b \in \{0, 1\}$ has been received: conditionally multicast $((\mathtt{prepare}, r, b, i), \mathsf{p})$, where p is a valid epoch-r prepare evidence for the bit b of the following form: if $r = 1$, a valid prepare evidence for b must be the sender's signature on b; else for $r > 1$, a valid prepare evidence for b is of the form $\mathsf{p} := (\mathtt{propose}, r, b, j)$ and moreover it must be that[a] $\mathcal{F}_{\text{mine}}.\mathtt{verify}((\mathtt{propose}, r, b), j) = 1$. If a valid proposal is received for both bits, pick either bit to send `prepare` for.
- **Commit.** If node i has observed at least $\mathcal{T}$ valid epoch-r prepare messages from distinct nodes (including itself) vouching for the same bit b, and moreover it has not received any valid epoch-r prepare message vouching for $1 - b$: node i conditionally multicasts the message $((\mathtt{commit}, r, b, i), \mathsf{POP}_b)$ where POP_b serves as an evidence to the commit message, and is formed by any set of $\mathcal{T}$ valid prepare messages of the form $((\mathtt{prepare}, r, b, \_), \_)$ from distinct nodes all vouching for b. POP_b is also said to be an epoch-r proof-of-preparation vouching for b.
- **Report.**
  - *Finalize.* If node i sees at least $\mathcal{T}$ valid `commit` messages from distinct nodes (including itself) for the same bit b: it finalizes and outputs b (if it has not output anything so far) and continues to participate in the protocol until all R epochs finish.
  - *Send report.* Every node i chooses a bit b such that a valid epoch-r proof-of-prepration (denoted POP_b) has been observed for b, and conditionally multicasts $((\mathtt{report}, r, b, i), \mathsf{POP}_b)$. If no such bit exists, conditionally multicasts $((\mathtt{report}, r, \perp, i), \perp)$. Note that an epoch-$r$ report for a non-$\perp$ bit $b \in \{0, 1\}$ is valid iff the report is attached with an evidence that is a proof-of-preparation for b.

[a] We repeat writing r and b here for convenience later when we instantiate $\mathcal{F}_{\text{mine}}$ with cryptography.

Figure 2: **Adaptively secure Byzantine broadcast: synchronous network with honest majority.** Described in the $\mathcal{F}_{\text{mine}}$-hybrid world. Appendix D explains how to instantiate $\mathcal{F}_{\text{mine}}$ using cryptography.

In any epoch r, if any so-far-honest node outputs a bit b in epoch r, then no so-far-honest node will ever observe a valid epoch-r proof-of-preparation for $1-b$.

To see the above, notice that a so-far-honest node outputs b in epoch r only if it has observed $f+1$ commit messages (from distinct nodes) vouching for b. One of these commit messages must have been sent by a forever-honest node henceforth indexed by i^*. Now node i^* could not have observed conflicting epoch-r prepare messages vouching for the different bit $1-b$. Suppose that at some time some honest node observes a valid epoch-r proof-of-preparation for $1-b$ denoted POP_{1-b}. This POP_{1-b} must contain $f+1$ prepare messages from distinct nodes for $1-b$, and one such prepare message must be from a forever-honest node — now this forever-honest must have multicast the prepare message to everyone; henceforth node i^* must have observed an epoch-r prepare message for $1-b$. Thus, we have reached a contradiction.

This critical observation leads to several useful conclusions:

1. *Consistency within an epoch.* First, observe that if two so-far-honest nodes output b and b' respectively in the same epoch r, it must be that $b = b'$.

2. *Consistency across epochs.* Consistency across epochs is ensured with the help of proposal evidence. If a so-far-honest node outputs b in epoch r, this node must have observed at least one commit message from a forever-honest node. Since this forever-honest node would have multicast the same commit message to everyone and moreover, no epoch-r commit message can contradict b, we may conclude that all forever-honest nodes (at least $f+1$ of them) will send an epoch-r report vouching for b. This means that in the next epoch $r+1$, any valid evidence must vouch for b. Thus, a valid proposal for epoch $r+1$ must propose b. In our formal proofs later, we will show that by induction, in fact, in all subsequent epochs greater than r, any valid proposal can only propose b. Note that this induction as well as the formal definition of valid evidence are arguably the most subtle part of the simplified protocol (without random eligibility election).

3. *Liveness.* Finally, we argue that by the end of epoch R, every forever-honest node will have output a bit. Observe that whenever a forever-honest node is the leader in some epoch r (and such an r must exist because $R \geq f+1$), it will sign a unique and valid proposal. It follows that in this epoch r, all nodes (that have not already produce an output earlier) will output some bit.

B.2.2 Achieving Small Bandwidth with Vote-Specific Eligibility Election

To achieve small bandwidth, we rely on vote-specific eligibility election:

- If all nodes were honest, then in expectation one node is elected to be a proposer every two epochs;
- If all nodes were honest, in expectation, $\lambda = \omega(\log \kappa)$ nodes would be elected to send prepare (or commit, report resp.) messages in every epoch.

Here, it is critical that *the committee/leader election be tied to each bit.* For example, following part of the consistency argument in Appendix B.2.1, we would like to guarantee that if a so-far-honest node has observed at least $\mathcal{T}$ prepare messages from distinct nodes all vouching for b in some epoch r (i.e., an epoch-r proof-of-preparation), then at least one of these prepare messages must be sent by a *so-far honest* node, and moreover this so-far-honest node must have sent the prepare message to everyone. Note that this argument is only possible if the committee for $(\texttt{prepare}, r, b =$

0) is elected independently from that for $(\texttt{prepare}, r, b = 1)$. Otherwise if $(\texttt{prepare}, r, b = 0)$ and $(\texttt{prepare}, r, b = 1)$ shared the same committee, then the adversary could immediately corrupt a node who has sent $(\texttt{prepare}, r, b = 0)$ and make it send $(\texttt{prepare}, r, b = 1)$ too (and therefore it could be that all prepare messages in a proof-of-preparation for $b = 1$ are from already corrupt nodes). A similar argument can be made for other types of messages, and we will give the formal proof in the next subsection.

For liveness, an important insight is that in $R = \lambda$ epochs, except with negligible probability, there must exist a "good" epoch in which exactly one so-far-honest node i^* is elected to propose a single bit b, and no already corrupt (including i^* if it becomes corrupt immediately after making the proposal) is elected as proposer. In such a good epoch, every forever-honest node is guaranteed to output a decision. Note that here again it is also important that the proposer election is bit-specific: otherwise, once a so-far-honest node is elected and proposes the bit 0, an adversary can corrupt it immediately and make it propose 1 too and in this case liveness will not ensue for this epoch.

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

In this section, we will prove the following theorem pertaining to the honest majority protocol described in Appendix B.1.

Theorem 4. *For any arbitrarily small positive constant ϵ, any $n \in \mathbb{N}$, the $\mathcal{F}_{\text{mine}}$-hybrid Byzantine broadcast protocol described in Figure 2 satisfies consistency, validity, and 4λ-termination (recall that λ may be any super-logarithmic function in the security parameter κ) in $(n, \frac{1}{2} - \epsilon)$-environments.*

Proof. As we shall see, the theorem follows from Lemmas 7 (consistency), 4 (validity) and 8 (liveness). □

Two types of bad events can contribute to the protocol's security being broken: 1) signature failure; and 2) stochastic bad events related to $\mathcal{F}_{\text{mine}}$'s random coins. In fact, only Lemma 4 for validity depends on the security of the signature scheme. In Figure 2, only the sender uses a signature scheme; hence, a compromised signature from the sender can be treated as a corrupt sender.

Assuming that the signature scheme is secure, the remainder of the section will focus on bounding the stochastic failures related to $\mathcal{F}_{\text{mine}}$'s random coins. The validity claim can be achieved by taking a union bound over the two types of bad events.

Recall that the number of nodes is n. We will conduct the analysis for $0 < \epsilon \leq \frac{1}{6}$ such that the adversary can (adaptively) corrupt at most $f := \lfloor (\frac{1}{2} - \epsilon)n \rfloor$ nodes. Observe that for $\epsilon > \frac{1}{6}$, one can still run the stronger protocol that pretends $\epsilon = \frac{1}{6}$.

The final choice of the failure probability δ due to $\mathcal{F}_{\text{mine}}$'s random coins is chosen to be $\exp(-\Theta(\lambda))$, which is negligibly small in the security parameter κ. However, in our intermediate lemmas, we sometimes overload δ to denote the failure probability of intermediate bad events.

We set the success probability of mining $\mathcal{F}_{\text{mine}}.\texttt{mine}(\cdot)$ a $\texttt{propose}$ message to $\frac{1}{n\Gamma}$, where Γ is the number of non-bottom possible values that b can take; for instance, $\Gamma = 2$ in the binary case.

Lemma 1 (Difficulty of Mining Propose Messages)**.** *Suppose $\mathcal{F}_{\text{mine}}.\texttt{mine}(\texttt{propose}, r, b)$ succeeds with $\frac{1}{n\Gamma}$ probability for a fixed epoch r and some b from Γ possible values.*

Then, for any epoch r, the probability that there is exactly one node and one value b with a successful $\mathcal{F}_{\text{mine}}.\texttt{mine}(\texttt{propose}, r, b)$ is in the range $[\frac{1}{e}, 0.43]$.

Proof. The required probability is $n\Gamma \cdot \frac{1}{n\Gamma}(1 - \frac{1}{n\Gamma})^{n\Gamma - 1}$, which has the desired range for $n \geq 2$ and $\Gamma \geq 2$. □

If a so-far-honest node calls $\mathcal{F}_{\text{mine}}.\texttt{mine}$ it is referred to as an *honest* mining attempt; if an already corrupt node calls $\mathcal{F}_{\text{mine}}.\texttt{mine}$ it is referred to as a *corrupt* mining attempt. In particular, if a node multicasts a message corresponding to a bit b, and then becomes corrupt in the same round and is immediately made to mine a message for $b' \neq b$, the latter treated as a corrupt mining attempt.

Lemma 2 (Choice of the Number R of Epochs)**.** *By choosing the $R = \Theta(\Gamma \log \frac{1}{\delta})$ as the number of epochs, except with probability δ, there is at least one epoch r in which exactly one honest mining attempt for a* `propose` *message is successful and no corrupt mining attempt for a* `propose` *message is successful.*

Proof. For each epoch r, fix some value value b such that the evidence from epochs up to $r-1$ can support value b.

From Lemma 1, the probability that in epoch r, there is exactly one node and exactly one value that can form a valid proposal is $\Theta(1)$. Hence, the probability that such a message is for value b and from a so-far-honest node (which is a constant fraction among all nodes) is $\Theta(\frac{1}{\Gamma})$.

Therefore, the probability that this fails to happen in all R epochs is at most $(1 - \Theta(\frac{1}{\Gamma}))^R \leq \exp(-\Theta(\frac{R}{\Gamma}))$. Choosing the specified value of R, the result follows. □

Lemma 3 (Difficulty of Mining Other Messages)**.** *Suppose the probability p of success for mining a message category from* $\{\texttt{prepare}, \texttt{commit}, \texttt{report}\}$ *(for each epoch r and each value b) is chosen to be $p := \Theta(\frac{1}{\epsilon^2 n} \log \frac{1}{\delta})$ such that $\mathcal{T} := \frac{np}{2}$ is an integer.*

(a) Fix $\texttt{T} \in \{\texttt{prepare}, \texttt{commit}, \texttt{report}\}$ *and an epoch r. The probability that in epoch r, there are less than $\mathcal{T}$ successful honest attempts to mine messages of category* `T` *(each node perhaps with a different value b) is at most δ.*

(b) Fix $\texttt{T} \in \{\texttt{prepare}, \texttt{commit}, \texttt{report}\}$*, an epoch r and some value b (that can be bottom). The following event holds except with probability at most δ: in epoch r, if there are at least $\mathcal{T}$ messages of category* `T` *for value b, at least one of the messages must be a successful honest attempt.*

Proof. We first prove statement (a). Observe that in any epoch, there are at least $(\frac{1}{2} + \epsilon)n$ so-far-honest node, each of which can successfully mine such a message in an epoch with probability p. By Chernoff Bound, the probability that there are less than $\mathcal{T}$ successes in some epoch is at most $\exp(-\Theta(\epsilon^2 np)) \leq \delta$.

We next consider statement (b). We fix an epoch r and some value b. **Important:** Observe that if any node (whether already corrupted adaptively or so-far-honest) has not attempted to mine a message of `msg_type` for value b in epoch r, then the success probability is p; moreover, the mining events for different nodes are independent.

Since there are at most $(\frac{1}{2} - \epsilon)n$ dishonest nodes at any moment, by Chernoff Bound, the probability that they can produce at least $\mathcal{T}$ successes is at most $\exp(-\Theta(\epsilon^2 np)) \leq \delta$.

□

Corollary 1 (Good Event)**.** *Suppose the probability p of success for mining a* `prepare`*,* `commit` *or* `report` *message (for each epoch r and each b from Γ possible values) is chosen to be $p := \Theta(\frac{1}{\epsilon^2 n} \log \frac{R\Gamma^2}{\delta})$ such that $\mathcal{T} := \frac{np}{2}$ is an integer.*

Then, except with probability at most δ, the following good event $\mathfrak{G}$ *happens. For every epoch r, and every* $\texttt{T} \in \{\texttt{prepare}, \texttt{commit}, \texttt{report}\}$*, we have:*

- *In epoch r, if every so-far-honest node attempts to mine a message of category* `T` *once (maybe each with a different value b), then at least $\mathcal{T}$ of them will be successful.*
- *For each value b, if there are at least $\mathcal{T}$ (valid) messages of category* `T` *for the same value b in epoch r from distinct nodes, then at least one of these messages is a successful honest attempt.*

Proof. By scaling the failure probability in Lemma 3 appropriately, applying the union bound over all epochs r and values b for all message types gives the result. □

Lemma 4 (Validity)**.** *Suppose the good event $\mathfrak{G}$ holds, with failure probability chosen to be $\delta = e^{-\Theta(\lambda)}$. Moreover, suppose the signature scheme fails with probability at most $\widehat{\delta}$. Then, except with probability $\delta + \widehat{\delta}$, the following holds: if the sender remains honest in the first epoch, then every so-far-honest node at the end of epoch 1 finalizes to the same value as the sender's value.*

Proof. Observe that in epoch 1, any valid `prepare` message must include the sender's signature; hence, except with $\widehat{\delta}$ probability, there can be `prepare` messages only for the proposed bit by the honest sender, by the security of the signature scheme.

The good event $\mathfrak{G}$ in Corollary 1 happens with probability at least $1-\delta$, which ensures that there are enough `prepare` and `commit` messages for all honest nodes to finalize to the same value. □

Lemma 5 (Consistency with Finalize)**.** *Suppose the good event $\mathfrak{G}$ in Corollary 1 happens. Moreover, suppose in some epoch r, a so-far-honest node finalizes to some value b. Then, the following holds.*

- *In epoch r, there is no valid* `commit` *message (from any node) for another different value b'.*
- *In epoch r, any valid* `report` *message is either for value b or $\perp$; moreover, there are at least $\mathcal{T}$* `report` *messages from so-far-honest nodes, and among any $\mathcal{T}$ valid* `report` *messages from distinct nodes, at least one is for value b.*

  In particular, this implies that in epoch $r+1$, the value b is the only valid value that can be proposed.

Proof. Suppose some so-far-honest node i finalizes to value b in epoch r.

We consider the first statement. Suppose, for contradiction's sake, there is a valid `commit` message for a different value b'. Since the so-far node i finalizes to value b, it must see at least $\mathcal{T}$ valid `commit` messages for value b, at least one of which must be multicast by some so-far-honest node $\widehat{i}$ (which could be the same as i), because the good event $\mathfrak{G}$ in Corollary 1 happens.

However, if there is a valid `commit` message for value b', then its evidence must contain $\mathcal{T}$ valid `prepare` messages from distinct nodes for value b', one of which must be from some so-far-honest node, due to the good event $\mathfrak{G}$. This means that every node, including node $\widehat{i}$, has seen a `prepare` message for value b'; therefore, the so-far-honest node $\widehat{i}$ should not have multicast a `commit` message for value b, reaching the desired contradiction. □

Lemma 6 (Persistent Value in Future Epochs)**.** *Suppose the good event $\mathfrak{G}$ from Corollary 1 holds. Suppose for some epoch r, in all epoch-r proposal evidence (from report messages up to epoch $r-1$) that exist in any so-far-honest node's view ever in* `view`*, b is the only bit that is vouched for.*

Then, in epoch r, any valid `commit` *message must be for value b; this implies that in epoch $r+1$, value b is the unique value that can appear in any valid* `propose` *message.*

Proof. Consider any valid `commit` message in epoch r, which must contain $\mathcal{T}$ valid `prepare` messages from distinct nodes, one of which must be so-far-honest, due to the good event $\mathfrak{G}$. Since, in any so-far-honest nodes' view, the only valid value that can be proposed in epoch r is b, it follows that

a so-far-honest will only multicast a `prepare` message for value b. It follows that any valid `commit` messages must be for value b in epoch r. This implies that any `report` message in epoch r must be for value b or $\perp$.

Combining with past `report` messages from earlier epochs, this implies that the value b remains the only value that can appear in any valid proposal in epoch $r + 1$. □

Lemma 7 (Consistency between Honest Finalizations)**.** *Suppose the good event $\mathfrak{G}$ from Corollary 1 holds. Then, if two so-far-honest nodes have finalized, they must finalize to the same value.*

Proof. For contradiction's sake, suppose two so-far-honest nodes finalizes to different (non-bottom) values $b \neq b'$.

Observe that for an honest node to finalize to some value in an epoch, Lemma 5 implies that all valid `commit` messages must be for value b in that epoch.

Therefore, two so-far-honest nodes cannot finalize to different values in the same epoch. Suppose one honest finalization to value b happens in an earlier epoch r.

However, Lemma 6 implies that from epoch $r + 1$ onwards, all valid `commit` messages must be for value b. This means that an honest node cannot finalize later to another value b'. This completes the proof. □

Lemma 8 (Liveness)**.** *Except with probability δ (chosen to be $e^{-\Theta(\lambda)}$), by the end of epoch R, all honest nodes will finalize to the same value.*

Proof. Scaling the failure probability appropriately in Corollary 1, except with probability at most $\frac{\delta}{2}$, the good event $\mathfrak{G}$ happens. By Lemma 7, no two honest nodes can finalize to different values. Hence, it suffices to prove that except with probability at most $\frac{\delta}{2}$, all honest nodes will finalize by the end of the protocol.

Observe that at the beginning of each epoch r, there exists at least one value b (for instance, as required by Lemma 6) such that if an honest node is eligible to propose b and no other node is eligible to propose, then the good event $\mathfrak{G}$ implies that all honest nodes will receive enough prepare and commit messages to finalize to b.

Applying Lemma 2 with failure probability $\frac{\delta}{2}$, there is some epoch in which such an honest node exists.

Hence, applying the union bound to all the aforementioned bad events, it follows that except with probability δ, all honest nodes finalize to the same value by the end of epoch R. □

Lemma 9 (Bandwidth Complexity)**.** *Except with probability at most* $\exp(-\Theta(Rnp)) = \exp(-\Theta(\lambda \log \lambda))$*, the multicast complexity is at most* $R \cdot O(np) = \Theta(\frac{\lambda \log \lambda}{\epsilon^2})$ *messages, where* $p := \Theta(\frac{1}{\epsilon^2 n} \log \frac{1}{\delta})$.

Proof. Observe that in each epoch, each honest node will try to mine at most one message of each category. The result follows from a standard application of Chernoff Bound. □

B.4 Early Termination

In the protocol in Figure 2, even though a node has finalized to some value, it still continues to participate in the protocol. We next augment the protocol such that it is possible that all honest nodes can terminate earlier. In particular, the expected number of epochs for all honest nodes to terminate is $O(1)$.

Pre-emptive mining. The intuition is that from Lemmas 5, once a (so-far-honest) node i finalizes to some value b because it has seen at least $\mathcal{T}$ epoch-r valid commit messages $\mathcal{M}$, then, with all but negligible probability, only the value b can be proposed after epoch r.

Hence, node i can pre-emptively mine all messages for value b for all epochs greater than $r+1$. In case, a mining is successful, that message will be multicast, using the set $\mathcal{M}$ of $\mathcal{T}$ messages as evidence.

Augmented Rule for Checking Messages. Upon receiving such a pre-mined message $((\mathtt{T}, r', b), i, \mathcal{M})$, another node can check its validity by verifying that node i has mined the message successfully, and its evidence $\mathcal{M}$ indeed corresponds to $\mathcal{T}$ epoch-r commit messages (for some $r < r'$) that can support the finalization to value b.

We describe the augmentation more formally as follows.

Parameters: The same threshold $\mathcal{T}$ as in the original protocol is used. If there is network delay, there is some upper bound estimate Δ.
Finalize. Suppose some node i finalizes to b for the first time, because it sees a set $\mathcal{M}$ of $\mathcal{T}$ epoch-r commit messages that support finalization to value b. Then, it performs the following.
Pre-emptive Mining. For each $r' \in [r+1..R]$, for each possible category $\mathtt{T}$ in the protocol, it conditionally multicasts the message $((\mathtt{T}, r', b), i, \mathcal{M})$.
Early Termination. Node i outputs b to the environment $\mathcal{Z}$, and terminates. Other nodes can use its pre-emptively mined messages for the rest of the protocol.

Figure 3: Early Termination.

Lemma 10. *The early termination augmentation maintains validity and consistency; and further, in $O(1)$ expected number of epochs all forever-honest nodes terminate.*

Proof. As explained in the intuition above, from Lemma 5, if a so-far-honest node i terminates with value b in some epoch r, then in all future epochs, only value b can be proposed, and b is the only non-bottom value that can be reported. Hence, in order to terminate early, it suffices for node i to mine, with the correct probabilities, all messages (vouching for bit b in future epochs) that might be needed for the protocol to continue. Indeed, pre-mining these messages ensures that the adversary cannot corrupt these future messages, which could happen without pre-mining. Hence, both validity and consistency follow as in the original protocol.

Furthermore, observe that if an honest node is the unique node that can successfully multicast a `propose` message (and the network delay is small enough for subsequently induced messages to take effect), then all honest node will terminate, as described in Lemmas 8.

Finally, since for each epoch, this happens with constant probability (by Lemmas 1), the expected number of epochs for this to happen is constant. □

Possible Further Optimization. The astute reader might observe that since all pre-mined messages from a terminating node are for the same value, it should be possible to have just one single pre-mined message that can be interpreted as a message of *universal* category which can be used in all future epochs. Indeed, we will use this idea in Appendix C.3, where the upper bound on the number of epochs depends on the network delay Δ, which might be unknown, but the reasoning will be more subtle.

B.5 Extension: Realizing the Agreement Version

So far we have focused on describing the broadcast version of the protocol. It is not too difficult to extend our results to the agreement version. Recall that in the broast version, in epoch $r = 1$,

there are the following cases:

1. *Forever-honest sender with input b*: in this case every honest node receives a prepare evidence for b in epoch $r = 1$ (i.e., a signature from the sender on b); and the adversary is unable to forge an epoch-1 prepare evidence for $1 - b$ except with negligible probability.

2. *Corrupt sender*: an honest node may or may not an epoch-1 prepare evidence for a bit; it may also receive an epoch-1 prepare evidence for both bits. In this case we do not care what happens.

Our idea is to introduce a small sub-protocol for nodes to disseminate their inputs before nodes enter the first epoch of the main protocol in Figure 2, and the initial sub-protocol will achieve the following (analogous to the above two cases):

1. *All forever-honest honest have the same input b*: in this case, we want that except with negligible probability, every honest node receives a prepare evidence for b in epoch $r = 1$; and that the adversary is unable to forge an epoch-1 prepare evidence for $1 - b$.

2. *Otherwise*: an honest node may or may not an epoch-1 prepare evidence for a bit; it may also receive an epoch-1 prepare evidence for both bits. In this case we do not care what happens.

An initial sub-protocol. The initial sub-protocol consists of two synchronous rounds, and the probability of successfully mining an `input` message on every attempt is the same as that of mining a `prepare`, `commit`, or `report` message (see Figure 2).

- Round 1: if an honest node i has the input bit 1, it conditionally multicasts $(\texttt{input}, 1, i)$; otherwise do nothing;

- Round 2: if an honest node i has heard at least $\mathcal{T}$ number of `input` messages vouching for the bit 1, do nothing; else, it conditionally multicasts $(\texttt{input}, 0, i)$;

- At the end of round 2: if the node has observed at least $\mathcal{T}$ `input` messages from distinct nodes for some bit b, output that set — henceforth such a set is considered a valid epoch-1 prepare evidence for the bit b. If at least $\mathcal{T}$ `input` messages have been observed for both bits, output either set.

Claim 3. *Except with negligible probability, the following holds: if all forever-honest nodes have the same input bit b, then every forever-honest node outputs a valid epoch-1 prepare evidence for b in the above initial sub-protocol; further, the adversary cannot forge an epoch-1 prepare evidence for* $1 - b$.

Proof. If all forever-honest nodes have the same input 1, applying Chernoff bound in a similar manner as before, we know that except with negligible probability, at the beginning of of round 2, all honest nodes will have seen at least $\mathcal{T}$ `input` messages vouching for 1. In this case, no honest node will attempt to conditionally multicast an `input` message for 0; and we conclude that except with negligible probability, the adversary cannot output a set of $\mathcal{T}$ `input` messages for 0.

Conversely, if all forever-honest nodes have the same input bit 0, then except with negligibility: 1) no honest node will try to conditionally multicast an `input` message for 1; 2) in round 2, no honest node can possibly have observed $\mathcal{T}$ `input` messages for 1; and thus 3) all honest nodes will attempt to conditionally multicast an `input` message for 0 — therefore at the end of the initial sub-protocol, all honest nodes will have observed at least $\mathcal{T}$ number of `input` messages for 0. □

Protocol for agreement. First, nodes run the initial sub-protocol described above. When the sub-protocol terminates after exactly 2 rounds, nodes enter epoch 1 of Figure 2. However, epoch 1 skips the **Propose** round and directly enters the **Prepare** round. If a node i's initial sub-protocol has output an epoch-1 prepare evidence (denoted E) for the bit b, then conditionally multicast $((\texttt{prepare}, 1, b, i), E)$.

It is not difficult to see that the consistency still holds for the new variant. Validity from the properties of the initial sub-protocol. Specifically, due to Claim B.5, the previous validity proof still holds by replacing signature security with Claim B.5.

C Partially Synchronous Network with $\frac{1}{3} - \epsilon$ Corruptions

In this section, we describe a partially synchronous BA protocol that tolerates $\frac{1}{3} - \epsilon$ adaptive corruptions and achieves sublinear multicast complexity without assuming erasures.

In this section, we describe our protocol in the $\mathcal{F}_{\text{mine}}$-hybrid world, and later in Appendix D, we will explain how to instantiate $\mathcal{F}_{\text{mine}}$ with cryptography while retaining the protocol's security properties.

C.1 Detailed Protocol

Earlier in Section 4, we gave an intuitive description of our partially synchronous protocol. Therefore in this section we directly jump into a formal presentation.

Message category and validity. All protocol messages are of the form

$$(\texttt{T}, r, b, i) \text{ or } ((\texttt{T}, r, b, i), E),$$

where $\texttt{T}$ is a category in $\{\texttt{propose}, \texttt{prepare}, \texttt{report}\}$, r denotes an epoch number, and $b \in \{0, 1\}$ if $\texttt{T} \in \{\texttt{propose}, \texttt{prepare}\}$ and $b \in \{0, 1, \perp\}$ for `report` messages.

Upon receiving any message of the above forms, a node would first check its validity by verifying:

- $\mathcal{F}_{\text{mine}}.\texttt{verify}((\texttt{T}, r, b), i) = 1$; and
- For `propose` or `report` messages, E must also be a valid supporting evidence. The definition of a valid evidence for these messages are deferred to Figure 4.

Conditionally multicast. Just like in Appendix B, we use the shorthand "conditionally multicast" to denote the fact that a node first mines a ticket with $\mathcal{F}_{\text{mine}}$ and, if successful, multicast a message.

Detailed protocol. The detailed protocol is presented in Figure 4 assuming an ideal functionality $\mathcal{F}_{\text{mine}}$. Although the activation points in each epoch look similar to the earlier Figure 2, we stress that here these activation points are not synchronous rounds one after another, instead they are triggered whenever some event or timeout is satisfied. Figure 4 does not deal with termination and honest nodes continue to participate forever even after they output a finalized bit to $\mathcal{Z}$. We explicitly deal with termination later in Appendix C.3.

Partially Synchronous BA Protocol with $\frac{1}{3} - \epsilon$ Corruptions

Parameters:

- Let λ be a super-logarithmic function in the security parameter κ.
- We set the mining difficulty parameters such that a propose message is successfully mined with $\frac{1}{2n}$ probability, and a commit or report message is successfully mined with $p := \Theta(\frac{\lambda}{\epsilon^2 n})$ probability.
- We set the threshold parameter $\mathcal{T} := \frac{2np}{3}$.
- Epoch r's deadline is set to be $D(r) := 2^{\lfloor \frac{r}{R} \rfloor} \cdot (R + (r \mod R))$ where $R = \lambda$, i.e., there are R epochs of length 1, followed by R epochs of length 2, followed by R epochs of length 4, and so on.

Initial report. Every node i conditionally multicasts $((\texttt{report}, 0, b_i, i), \perp)$ where b_i is node i's input.

Epoch $r = 1, 2, \ldots$**.** In epoch r, each node i performs the following.

- **Propose**.
  - The *first time* a node i has observed a valid evidence E (from `report` messages sent before epoch r) for proposing $(\texttt{propose}, r, b)$ for some bit $b \in \{0, 1\}$, it conditionally multicasts $((\texttt{propose}, r, b, i), E)$.
  - E is said to be a valid proposal evidence for $(\texttt{propose}, r, b)$ where $b \in \{0, 1\}$ and $r \geq 1$, iff E contains, for every $0 \leq r' < r$, a set of $\mathcal{T}$ valid report messages of the form $((\texttt{report}, r', \texttt{b}_i, i), \mathsf{POP}_i)$ from distinct nodes such that the following rules are respected:

    1) If at least one of the report messages from epoch $r' \geq 1$ in E contains a non-$\perp$ value $\texttt{b}_i$, then b must be one of the non-$\perp$ values with the maximum epoch number r'; **else** consider the next rule.

    2) If, for $r' = 0$, the epoch-0 report messages in E contain a **strong majority** of at least $\lfloor \frac{\mathcal{T}}{2} \rfloor + 1$ messages for the same value, then b must be this value.

    3) If both of the following rules cannot determine a value, then b can be any non-$\perp$ value.
- **Prepare.** In each epoch, each node multicasts at most one `prepare` message as follows: upon receiving the *first* valid proposal of the form $((\texttt{propose}, r, b, \_), \_)$ for some bit $b \in \{0, 1\}$, it conditionally multicasts $(\texttt{prepare}, r, b, i)$. Any subsequent epoch-$r$ `propose` messages will be ignored.
- **Report**. Wait till the epoch deadline $D(r)$. If by $D(r)$, node i has received at least $\mathcal{T}$ valid prepare messages of the form $(\texttt{prepare}, r, b, i)$ from distinct nodes for the same $b \in \{0, 1\}$, it does the following:
  - Pick a collection of $\mathcal{T}$ prepare messages to form a proof-of-preparation for b, denoted POP_b.
  - Conditionally multicast $((\texttt{report}, r, b, i), \mathsf{POP}_b)$.

  Else, conditionally multicast $((\texttt{report}, r, \perp, i), \perp)$.

  Henceforth, POP_b is said to be a valid evidence for $(\texttt{report}, r, b)$ where $r \geq 1$ iff either $b = \perp$ or POP_b is a valid epoch-r proof-of-preparation for b.

Finalize. Any any time in the protocol, if $\mathcal{T}$ valid report messages of the form $((\texttt{report}, r, b, \_), \_)$ for the same $b \in \{0, 1\}$ and the same epoch r from distinct nodes have been received, a node finalizes and outputs b to $\mathcal{Z}$ and continue participating in the protocol.

(Early) Termination. For this description, all nodes continue to participate forever even after they have output a bit to $\mathcal{Z}$. In Appendix C.3, we describe how a node can *terminate*.

Figure 4: **Adaptively secure partially synchronous BA protocol.** The protocol is described in the $\mathcal{F}_{\text{mine}}$-hybrid world. Appendix D will explain how to instantiate $\mathcal{F}_{\text{mine}}$ with cryptographic assumptions.

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

To aid understand, an informal proof overview was presented earlier in Section 4. In this section, we will focus on *formally* proving the following theorem pertaining to the partially synchronous protocol described in Appendix C.1.

Theorem 5. *For any arbitrarily small positive constant ϵ, any $n \in \mathbb{N}$, the $\mathcal{F}_{\text{mine}}$-hybrid BA protocol described in Figure 4 satisfies consistency, validity, and $\Theta(\lambda\Delta)$-termination in $(n, \frac{1}{3} - \epsilon)$-environments.*

Note also that our $\mathcal{F}_{\text{mine}}$-hybrid protocol for partial synchrony (Figure 4) does not use any cryptographic primitives and therefore the above theorem holds for even unbounded $\mathcal{A}$ and $\mathcal{Z}$. Of course, later in Appendix D, when we actually instantiate $\mathcal{F}_{\text{mine}}$ with cryptography, the resulting real-world protocol would then be secure only against polynomially bounded $\mathcal{A}$ and $\mathcal{Z}$.

To complete the description and the analysis of the protocol, we specify the values of the parameters. Recall that n is the number of nodes, and $0 < \epsilon \leq \frac{1}{6}$ such that the adversary can (adaptively) corrupt at most $f := \lfloor (\frac{1}{3} - \epsilon)n \rfloor$ nodes. Moreover, $\delta := e^{-\Theta(\lambda)} > 0$ is the total failure probability of the protocol; however, in our intermediate lemmas, we sometimes overload δ to denote the failure probability of intermediate bad events.

We set the success probability of mining $\mathcal{F}_{\text{mine}}.\texttt{mine}(\cdot)$ a `propose` message to $\frac{1}{n\Gamma}$, where Γ is the number of non-bottom possible values that b can take; for instance, $\Gamma = 2$ in the binary case.

Lemma 11 (Difficulty of Mining Propose Messages)**.** *Suppose that $\mathcal{F}_{\text{mine}}.\texttt{mine}(\texttt{propose}, r, b)$ is successful with $\frac{1}{n\Gamma}$ probability for a fixed epoch r and some b from Γ possible values.*

Then, for any epoch r, the probability that there is exactly one node and one value b with a successful $\mathcal{F}_{\text{mine}}.\texttt{mine}(\texttt{propose}, r, b)$ is in the range $[\frac{1}{e}, 0.43]$.

Proof. The proof is the same as Lemma 1. □

Lemma 12 (Choice of the Parameter R)**.** *Choose $R = \Theta(\Gamma \log \frac{1}{\delta})$. For any R consecutive epochs, except with probability δ, there is at least one epoch r in which exactly one honest mining attempt for a* `propose` *message is successful and no corrupt mining attempt for a* `propose` *message is successful.*

Proof. The proof is the same as Lemma 2. □

Lemma 13 (Difficulty of Mining Other Messages)**.** *Suppose the probability p of success for mining a prepare or report message (for each epoch r and each value b) is chosen to be $p := \Theta(\frac{1}{\epsilon^2 n} \log \frac{1}{\delta})$ such that $\mathcal{T} := \frac{2np}{3}$ is an integer. Moreover, recall that in each epoch, a so-far-honest node will attempt to mine a prepare or report message at most once (for a single value). Then, the following holds.*

(a) Fix an epoch r and a category $\texttt{T} \in \{\texttt{prepare}, \texttt{report}\}$. The probability that in epoch r, less than $\mathcal{T}$ so-far-honest can successfully mine messages of category $\texttt{T}$ is at most δ.

(b) Fix an epoch r, a category $\texttt{T} \in \{\texttt{prepare}, \texttt{report}\}$ and distinct values b_1 and b_2 (that can be bottom). The probability of the following event is at most δ: in epoch r, for each of the values b_1 and b_2, there are at least $\mathcal{T}$ messages of category $\texttt{T}$ for that value that are successfully mined.

Proof. We first prove statement (a). Observe that there are at least $(\frac{2}{3}+\epsilon)n$ so-far-honest node, each of which can successfully mine a message of category `T` in an epoch with probability p. By Chernoff Bound, the probability that there are less than $\mathcal{T}$ successes in some epoch is at most $\exp(-\Theta(\epsilon^2 np)) \leq \delta$.

We next consider statement (b). We fix an epoch r and two distinct values b_1 and b_2. Suppose for each of the value, there are at least $\mathcal{T}$ successfully mined messages. This means there exist $2\mathcal{T} = \frac{4np}{3}$ such messages (for either b_1 or b_2) mined in epoch r. Observe that a so-far-honest node will try to mine such a message at most once, while a corrupted node can try to mine messages for both b_1 and b_2.

Since there are at most $(\frac{1}{3}-\epsilon)n$ corrupted nodes, the total number of attempts to mine for either b_1 or b_2 is at most $(\frac{4}{3}-\epsilon)n$, each of which has success probability p.

Therefore, by Chernoff Bound, the probability that there are $2\mathcal{T}$ successes is at most $\exp(-\Theta(\epsilon^2 np)) \leq \delta$, by the choice of p. □

Corollary 2 (Difficulty of Mining `prepare`)**.** *Suppose the probability p of success for mining a* `prepare` *message (for each epoch r and each b from Γ possible values) is chosen to be $p := \Theta(\frac{1}{\epsilon^2 n}\log\frac{R\log\Delta\cdot\Gamma^2}{\delta}) = \Theta(\frac{\lambda}{\epsilon^2 n})$ such that $\mathcal{T} := \frac{2np}{3}$ is an integer. Then, the following holds.*

(a) The probability that there exists some epoch r in which less than $\mathcal{T}$ so-far-honest can successfully mine `prepare` *messages is at most δ.*

(b) The probability that there exists some epoch r in which two POPs for two distinct values are produced is at most δ.

Proof. Both statements can be proved using Lemma 13. Statement (a) follows from an application of the union bound over R epochs. Statement (b) involves the union bound over all $\Theta(R\log\Delta)$ epochs and all $\Theta(\Gamma^2)$ pairs of distinct values, because, as we shall see in Lemma 19, the number epochs needed is at most $\Theta(R\log\Delta)$ with all but negligible probability. □

Lemma 14 (Consistency with Finalize)**.** *Suppose the probability p of success for mining a* `report` *message for each epoch r and each (possibly bottom) value b is chosen to be $p := \Theta(\frac{\lambda}{\epsilon^2 n})$ such that $\mathcal{T} := \frac{2np}{3}$ is an integer.*

Suppose in some epoch r, any two POPs must be for the same (non-bottom) value b (see Corollary 2 (b)).

Then, except with probability $\frac{\delta}{R}$, the following event happens in epoch r: if some so-far-honest node i has made an epoch-r induced finalization to value b, then any $\mathcal{T}$ epoch-r `report` *messages from distinct nodes must have b as the only non-bottom reported value, which implies that any valid* `propose` *message in epoch $r+1$ is for value b.*

Proof. By the hypothesis, b is the only non-bottom value that can appear in a POP; hence, any `report` message can be for the value b or $\perp$. Therefore, it suffices to give an upper bound $\frac{\delta}{R}$ for the probability of the following event: some so-far-honest node i has made an epoch-r induced finalization to value b (because it sees $\mathcal{T}$ report messages from distinct nodes for b), and there are $\mathcal{T}$ report messages from distinct nodes for $\perp$. However, Lemma 13 states that this happens with probability at most $\frac{\delta}{R}$, from the choice of p. □

Lemma 15 (Persistent Value in Future Epochs)**.** *Suppose for some epoch r, in all epoch-r proposal evidence (from report messages up to epoch $r-1$) that exist in any honest node's view ever in* view*, b is the only bit that is vouched for. Then, except with probability $\frac{\delta}{R}$, any valid POP from epoch r must be for value b, which implies that in epoch $r+1$, value b is the unique value that can appear in any valid* `propose` *message.*

Proof. To have a valid POP for another value b', there must be at least $\mathcal{T} = \frac{2np}{3}$ prepare messages for the value b'. However, there are at most $(\frac{1}{3} - \epsilon)n$ corrupted nodes that will try to mine `prepare` for another value b'. For any fixed b', by Chernoff Bound, the probability that there are at least $\mathcal{T}$ prepare messages for b' is at most $\exp(-\Theta(np)) \leq \frac{\delta}{\Gamma R}$. Applying the union bound over all values $b' \neq b$ gives the desired upper bound.

Next, observe that if there is no POP for $b' \neq b$ means that there cannot be any valid report for $b' \neq b$. Hence, it follows that in epoch r, each report is for value b or $\perp$. Therefore, combining with past reports from earlier epochs, the value b remains the only value that can appear in any valid proposal in epoch $r + 1$. □

Lemma 16 (Consistency between Honest Finalizations)**.** *Except with probability δ, the following happens: if any two so-far-honest nodes have both finalized, they must finalize with the same value.*

Proof. Using Corollary 2, Lemmas 14 and 15, by scaling δ appropriately, we can conclude that except with probability δ, the following event happens.

For any epoch $r \in [R]$, all of the following happen:

- Any two POPs in epoch r must be for the same (non-bottom) value.
- If some so-far-honest node has made an epoch r induced finalization, then b is the only (non-bottom) value that can be proposed in epoch $r+1$ (with respect to past `report` messages up to epoch r); moreover, from epoch $r + 1$ onwards, only POP with value b can be produced.

We show that if the above event happens, then it is impossible to have two so-far-honest nodes finalizing to different (non-bottom) values $b \neq b'$. For contradiction's sake, suppose the contrary happens. Observe that for a so-far-honest node to finalize with some value due to a POP in some epoch, then any POP in that epoch must be for the same value.

The above event implies that in an epoch, there cannot be two POPs for different values. Therefore, two finalizations to different values cannot be induced in the same epoch.

Suppose the finalization to value b is induced by an earlier epoch r. However, the above event implies that from epoch $r + 1$ onwards, only POPs for value b can be produced, which means a so-far-honest node will not be induced to finalize to another value b' due to a later epoch. This completes the proof. □

Lemma 17 (Liveness)**.** *Except with probability δ, by epoch at most $r = R \log_2 \Delta + O(R)$, all honest nodes will have finalized with the same value. In particular, by round number at most $O(R\Delta)$, all messages in epoch r will have reached their recipients.*

Proof. Observe that since the network delay is at most Δ, after epoch number $r \geq R \log_2 \Delta + O(R)$, all valid messages will reach their recipients in order to take effect before the corresponding break times. Specifically, before $D(r)$, all the following have happened: (1) every node receives enough report messages from up to epoch $r - 1$ in case it needs to multicast a propose message in epoch r; (2) any valid propose or prepare message from epoch r has reached its intended recipient.

Applying Lemma 16 with failure probability $\frac{\delta}{3}$, we conclude that no so-far two honest nodes can finalize with different values.

Observe that at the beginning of each epoch r, there exists at least one value b such that if a so-far-honest node is eligible to propose b and no other node is eligible to propose, then except with probability at most $\Theta(\frac{\delta}{R})$ (by Lemma 13 (a)), all so-far-honest nodes will receive enough prepare and report messages to finalize to value b, because the network delay is short enough after epoch number $R \log_2 \Delta + O(R)$.

Applying Lemma 12 with failure probability $\frac{\delta}{3}$, there is some epoch in which such a so-far-honest node exists.

Hence, applying the union bound to all the aforementioned bad events, it follows that except with probability δ, all honest nodes finalize to the same value. □

Lemma 18 (Validity)**.** *Suppose all nodes initially have the same input value b (out of $\Gamma = 2$ possible values).*

Suppose that the probability p of success for mining a `report` *message for each epoch 0 and each value b is chosen to be $p := \Theta(\frac{1}{\epsilon^2 n} \log \frac{2}{\delta})$ such that $\mathcal{T} := \frac{2np}{3}$ is an integer. Then, except with probability δ, at least $\mathcal{T}$ epoch-0 report messages are successfully mined from distinct so-far-honest nodes; furthermore, among any $\mathcal{T}$ epoch-0 report messages from distinct nodes, a strong majority of them must be for value b.*

In particular, the above event implies that only value b can be proposed in all epochs.

Proof. Suppose all nodes have the same input value b. Then, Lemma 13 implies that there are at least $\mathcal{T}$ report messages (for value b) from so-far-honest nodes except with probability at most $\frac{\delta}{2}$.

For the other value $b' \neq b$, there are at most $(\frac{1}{3} - \epsilon)n$ corrupt nodes that can try to mine `report` messages for b'. By Chernoff Bound, the probability that they can produce at least $\frac{\mathcal{T}}{2} = \frac{np}{3}$ such messages is at most $\frac{\delta}{2}$.

An application of the union bound over the above bad events gives the result. □

C.3 (Early) Termination

In Figure 4, even though a node has finalized to some value, it still continues to participate in the protocol (forever). We next augment the protocols such that it is possible that all honest nodes can terminate. In particular, in the augmented protocol, the expected number of epochs for all honest nodes to terminate is $R \log_2 \Delta + O(R)$, where $\Delta \geq 1$ is the network delay.

Pre-emptive mining. The intuition is that from Lemma 14, once a (so-far-honest) node i finalizes to some value b because it has seen at least $\mathcal{T}$ epoch-r valid `report` messages $\mathcal{M}$ then, with all but negligible probability, only the value b can be proposed after epoch r.

Hence, as in Appendix B.4, node i can pre-emptively mine all messages for value b for all epochs greater than $r + 1$. The issue is that we do not know for how many epochs in the future a node need to pre-mine messages, because there is no termination condition in Figure 4.

However, combining Corollary 2 and Lemma 14, one can observe that, as long as there exists $\mathcal{T}$ valid epoch-r report messages for value b, only the value b can be proposed after epoch r. Therefore, it should be possible to for a terminating node to just pre-mine one single message of some *universal* category that can be used in future epochs.

The `universal` category. We introduce a new message category known as `universal`, whose mining rules and probability are identical to `report`. In particular, the message has the form $((\texttt{universal}, r, b, i), \mathcal{M})$, where a valid evidence $\mathcal{M}$ is a collection of $\mathcal{T}$ epoch r report messages for value b, which supports finalization to value b.

Augmented Rule for Checking Messages. Upon receiving such a pre-mined message $((\texttt{universal}, r, b, i), \mathcal{M})$, another node can check its validity by verifying that node i has mined the message successfully, and its evidence $\mathcal{M}$ indeed corresponds to $\mathcal{T}$ epoch-r report messages that can support the finalization to value b. This message will be treated as a valid propose, prepare or report message for all epochs $r' \geq r + 1$.

We describe the augmentation more formally as follows.

Parameters: The same threshold $\mathcal{T}$ as in the original protocol is used.
Finalize. Suppose some node i finalizes to b for the first time, because it sees a set $\mathcal{M}$ of $\mathcal{T}$ epoch-r report messages for value b. Then, it performs the following.
Pre-emptive Mining. Node i conditionally multicasts the message $((\texttt{universal}, r, b, i), \mathcal{M})$.
Early Termination. Node i outputs b to the environment $\mathcal{Z}$, and terminates. Other nodes can use this pre-emptively mined message $((\texttt{universal}, r, b, i), \mathcal{M})$; for each $r' \geq r + 1$ and $\texttt{T} \in \{\texttt{propose}, \texttt{prepare}, \texttt{report}\}$, this pre-mined message will be treated as a valid epoch-r' message of category $\texttt{T}$.

Figure 5: Early Termination.

Lemma 19. *The termination augmentation maintains validity and consistency; and further, with all but negligible probability, in $R \log_2 \Delta + O(R)$ number of epochs, all forever-honest nodes terminate.*

Proof. As explained in the intuition above, validity and consistency follow from Corollary 2 and Lemma 14. To be more specific, they imply that once there are $\mathcal{T}$ valid epoch-r report messages $\mathcal{M}$ for the same value b, then, with all but negligible probability, then all valid propose and report messages in epoch $r' \geq r + 1$ must be for value b.

Hence, if a node is induced to finalize by the report messages $\mathcal{M}$, it can terminate as long as it has pre-mined all messages (propose, prepare, report) for value b with the correct probabilities that might be needed in the future epochs.

The subtlety here is that it just pre-mines a single universal message. Observe that in our proofs in Corollary 2 and Lemma 14, we only need independence among different nodes mining for messages of the same category in the same epoch (but perhaps for different values for corrupt nodes). It is important that we do not need independence among messages of different categories and different epochs for the same node, as long as we can be sure that the node remains honest, and this is achieved exactly by a universal message from a so-far-honest node.

Lemma 20 (Number of Messages)**.** *Except with probability at most* $\exp(-\Theta(R \log_2 \Delta \cdot np)) + \delta \leq e^{-\Theta(\lambda)}$*, the multicast complexity is at most $R \log_2 \Delta \cdot O(np)$ messages.*

Proof. Observe that in each epoch, each so-far-honest node will try to mine at most one message of each category. From Lemma 17, except with probability δ, the protocol terminates after epoch number $\Theta(R \log_2 \Delta)$.

The result follows from a standard application of Chernoff Bound and a union bound of the bad events from the Chernoff bound and Lemma 17. □

Multicast complexity. Due to Lemma 19 and since Δ must be polynomially bounded in κ, it is not difficult to see that our $\mathcal{F}_{\text{mine}}$-hybrid partially synchronous protocol with the early termination augmentation achieves $\mathsf{poly}\log(\kappa)$ multicast complexity. Later in Appendix D, when the protocol is instantiated in the real world replacing $\mathcal{F}_{\text{mine}}$ with cryptography, the multicast complexity will become $\mathsf{poly}\log(\kappa) \cdot \chi$ where χ is a separate security parameter related to the strength of the cryptographic assumptions.

□

D Instantiating $\mathcal{F}_{\text{mine}}$ in the Real World

So far, all our protocols 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 (where $\mathcal{F}_{\text{mine}}$ does not exist) using cryptography. Technically we do not directly realize the ideal functionality $\mathcal{F}_{\text{mine}}$ in the sense of Canetti [5] — instead, we describe a real-world protocol that preserves all the security properties of the $\mathcal{F}_{\text{mine}}$-hybrid protocols.

D.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,

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

D.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 6 (Instantiation of our NIZK and commitment schemes [18])**.** *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. [18] 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. □

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

D.4 Compiler from $\mathcal{F}_{\mathrm{mine}}$-Hybrid Protocols to Real-World Protocols

Our real-world protocols will remove the $\mathcal{F}_{\mathrm{mine}}$ 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.

Earlier in Section 1, we have described the intuition behind our approach. Hence in this section we directly provide a formal description of how to compile our $\mathcal{F}_{\mathrm{mine}}$-hybrid protocols into real-world protocols using cryptography. This compilation works for all of our previous $\mathcal{F}_{\mathrm{mine}}$-hybrid protocols described in Figures 2 and 4 respectively.

- **Trusted 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 $\mathcal{F}_{\text{mine}}.\texttt{mine}$.** Recall that in the ideal-world protocol a node i calls $\mathcal{F}_{\text{mine}}.\texttt{mine}(\mathsf{m})$ to mine a vote for a message m. Now, instead, the node i calls $\rho := \mathsf{PRF}_{\mathsf{sk}_i}(\mathsf{m})$, and computes the NIZK proof

$$\pi := \mathsf{nizk.P}((\rho, \mathsf{pk}_i, \mathsf{crs}_{\text{comm}}, \mathsf{m}), (\mathsf{sk}_i, s_i))$$

  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 m is considered successful if $\rho < D_p$ where D_p is an appropriate difficulty parameter such that any random string of appropriate length is less than D_p with probability p — recall that the parameter p is selected in a way that depends on the message m being "mined".

- **New message format.** Recall that earlier in our $\mathcal{F}_{\text{mine}}$-hybrid protocols, every message multicast by a so-far-honest node i must of one of the following forms:

  - Mined messages of the form (m, i) where node i has successfully called $\mathcal{F}_{\text{mine}}.\texttt{mine}(\mathsf{m})$; For example, in the synchronous honest majority protocol (Figure 2), m of be form $(\mathtt{T}, r, b)$ where $\mathtt{T} \in \{\texttt{propose}, \texttt{prepare}, \texttt{commit}, \texttt{report}\}$, r denotes an epoch number, and $b \in \{0, 1, \perp\}$.
  - Signed messages of the form (m, i, σ) where σ is a valid signature or some relevant evidence on m from i;
  - Compound messages, i.e., a concatenation of the above types of messages.

  Signed messages (that are either stand-alone or contained in other messages) need not be transformed in the real world. For every *mined* message (m, i) that is either stand-alone or contained in a compound message, in the real-world protocol, we rewrite (m, i) as $(\mathsf{m}, i, \rho, \pi)$ where the terms ρ and π are defined in the most natural manner:

  - If (m, i) is part of a message that a so-far-honest node i wants to multicast, then the terms ρ and π are those generated by i in place of calling $\mathcal{F}_{\text{mine}}.\texttt{mine}(\mathsf{m})$ in the real world (as explained above);
  - Else, if (m, i) is part of a message that a so-far-honest node $j \neq i$ wants to multicast, it must be that j has received a valid real-world tuple $(\mathsf{m}, i, \rho, \pi)$ where validity will be defined shortly, and thus ρ and π are simply the terms contained in this tuple.

- **Instantiating $\mathcal{F}_{\text{mine}}.\texttt{verify}$.** In the ideal world, a node would call $\mathcal{F}_{\text{mine}}.\texttt{verify}$ to check the validity of mined messages upon receiving them (possibly contained in compound messages). In the real-world protocol, we perform the following instead: upon receiving the mined message $(\mathsf{m}, i, \rho, \pi)$ that is possibly contained in compound messages, a node can verify the message's validity by checking:

  1. $\rho < D_p$ where p is an appropriate difficulty parameter that depends on the type of the mined message; and

2. π is indeed a valid NIZK for the statement formed by the tuple $(\rho, \mathsf{pk}_i, \mathsf{crs}_{\text{comm}}, \mathsf{m})$. The tuple is discarded unless both checks pass.

D.5 Main Theorems for Real-World Protocols

After applying the above compiler to our $\mathcal{F}_{\text{mine}}$-hybrid protocols described in Figures 2 and 4, we obtain our real-world protocols for the following settings respectively: 1) synchronous network with $\frac{1}{2} - \epsilon$ fraction corrupt, and 2) partially synchronous network with $\frac{1}{3} - \epsilon$ fraction corrupt. In this section, we present our main theorem statements for these three settings. The proofs for these theorems can be derived by combining the analyses in Appendices B.3 and C.2 as well as those in the following section, i.e., Appendix E where will show that the relevant security properties are preserved in the real world as long as the cryptographic building blocks are secure.

Henceforth in theorem statements, when we say that "*assume that the cryptographic building blocks employed are secure*", we formally mean that 1) the pseudorandom function family employed is secure; 2) the non-interactive zero-knowledge proof system that satisfies non-erasure computational zero-knowledge and perfect knowledge extraction; 3) the commitment scheme is computationally hiding under selective opening and perfectly binding; and 4) the signature scheme is secure (if relevant).

Theorem 7 (Synchronous network with honest majority)**.** *Let $\pi_{\text{honestmaj}}$ be the protocol obtained by applying the above compiler to Figure 2, and assume that the cryptographic building blocks employed are secure. Then, for any arbitrarily small positive constant ϵ, any $n \in \mathbb{N}$, $\pi_{\text{honestmaj}}$ satisfies consistency, validity, and $\mathsf{poly}\log(\kappa)$-termination in $(n, \frac{1}{2} - \epsilon)$-environments for a suitable polynomial function $\mathsf{poly}(\cdot)$. Further, $\pi_{\text{honestmaj}}$ achieves $\chi \cdot \mathsf{poly}\log(\kappa)$ multicast message complexity where χ is a security parameter related to the hardness of the cryptographic building blocks, and $\mathsf{poly}(\cdot)$ denotes another suitable polynomial function.*

Proof. The proof of this theorem can be obtained by combining the $\mathcal{F}_{\text{mine}}$-hybrid analysis in Appendix B.3 as well as Appendix E where we show that the relevant security properties are preserved in by the real world protocol. □

Theorem 8 (Partially synchronous network with 1/3 corrupt)**.** *Let $\pi_{\text{partialsync}}$ be the protocol obtained by applying the above compiler to Figure 4, and assume that the cryptographic building blocks employed are secure. Then, for any arbitrarily small positive constant ϵ, any $n \in \mathbb{N}$, $\pi_{\text{partialsync}}$ satisfies consistency, validity, and $(\Delta \cdot \mathsf{poly}\log(\kappa))$-termination in $(n, \frac{1}{3} - \epsilon)$-environments for a suitable polynomial function $\mathsf{poly}(\cdot)$. Further, $\pi_{\text{partialsync}}$ achieves $\chi \cdot \mathsf{poly}\log(\kappa)$ multicast message complexity where χ is a security parameter related to the hardness of the cryptographic building blocks, and $\mathsf{poly}(\cdot)$ denotes another suitable polynomial function.*

Proof. The proof of this theorem can be obtained by combining the $\mathcal{F}_{\text{mine}}$-hybrid analysis in Appendix C.2 as well as Appendix E where we show that the relevant security properties are preserved in by the real world protocol. □

E Real World is as Secure as the $\mathcal{F}_{\text{mine}}$-Hybrid World

E.1 Preliminary: PRF's Security Under Selective Opening

Our proof will directly rely on the security of a PRF under selective opening attacks. We will prove that any secure PRF family is secure under selective opening with a polynomial loss in the security.

Pseudorandomness under selective opening. We consider a selective opening adversary that interacts with a challenger. The adversary can request to create new PRF instances, query existing instances with specified messages, selectively corrupt instances and obtain the secret keys of these instances, and finally, we would like to claim that for instances that have not been corrupt, the adversary is unable to distinguish the PRFs' evaluation outcomes on any future message from random values from an appropriate domain. More formally, we consider the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

$\underline{\mathsf{Expt}_b^{\mathcal{A}}(1^\kappa)}$:

- $\mathcal{A}(1^\kappa)$ can adaptively interact with $\mathcal{C}$ through the following queries:
  - *Create instance.* The challenger $\mathcal{C}$ creates a new PRF instance by calling the honest $\mathsf{Gen}(1^\kappa)$. Henceforth, the instance will be assigned an index that corresponds to the number of "create instance" queries made so far. The i-th instance's secret key will be denoted sk_i.
  - *Evaluate.* The adversary $\mathcal{A}$ specifies an index i that corresponds to an instance already created and a message m, and the challenger computes $r \leftarrow \mathsf{PRF}_{\mathsf{sk}_i}(\mathsf{m})$ and returns r to $\mathcal{A}$.
  - *Corrupt.* The adversary $\mathcal{A}$ specifies an index i, and the challenger $\mathcal{C}$ returns sk_i to $\mathcal{A}$ (if the i-th instance has been created).
  - *Challenge.* The adversary $\mathcal{A}$ specifies an index i^* that must have been created and a message m. If $b = 0$, the challenger returns a completely random string of appropriate length. If $b = 1$, the challenger computes $r \leftarrow \mathsf{PRF}_{\mathsf{sk}_{i^*}}(\mathsf{m})$ and returns r to the adversary.

We say that $\mathcal{A}$ is compliant iff with probability 1, the challenge tuple (i^*, m) satisfies the following: 1) $\mathcal{A}$ does not make a corruption query on i^* throughout the game; and 2) $\mathcal{A}$ does not make any evaluation query on the tuple (i^*, m).

Definition 3 (Selective opening security of a PRF family)**.** *We say that a PRF scheme satisfies pseudorandomness under selective opening iff for any compliant p.p.t. adversary $\mathcal{A}$, its views in $\mathsf{Expt}_0^{\mathcal{A}}(1^\kappa)$ and $\mathsf{Expt}_1^{\mathcal{A}}(1^\kappa)$ are computationally indistinguishable.*

Theorem 9. *Any secure PRF family satisfies pseudorandomness under selective opening by Definition 3 (with polynomial loss in the security reduction).*

The proof of this theorem is standard and deferred to Appendix F.

E.2 Definition of Polynomial-Time Checkable Stochastic Bad Events

In all of our $\mathcal{F}_{\text{mine}}$-hybrid protocols earlier, there are two types of failiures that can lead to the breach of protocol security (i.e., consistency, validity, or termination):

1. Signature failure; and
2. Stochastic bad events related to $\mathcal{F}_{\text{mine}}$'s random coins. These stochastic bad events are of the form imprecisely speaking: either there are too few honest mining successes or there are too many corrupt mining successes. More formally, for the honest majority protocol, the stochastic bad events are stated in Lemma 2 and Corollary 1; and for the partially synchronous protocol, the stochastic bad events are stated in Lemma 12 and 13.

We point out that for the second category, i.e., stochastic bad events, for every protocol, there is a polynomial-time predicate henceforth denoted F, that takes in 1) all honest and corrupt mining attempts and the rounds in which the attempts are made (for a fixed view) and 2) $\mathcal{F}_{\text{mine}}$'s coins as a result of these mining attempts, and outputs 0 or 1, indicating whether the bad events are true for this specific view.

In our earlier $\mathcal{F}_{\text{mine}}$-world analyses (Appendices B.3 and C.2), although we have not pointed out this explicitly, but our proofs actually suggest that the stochastic bad events defined by F happen with small probability *even when $\mathcal{A}$ and $\mathcal{Z}$ are computationally unbounded*.

The majority of this section will focus on bounding the second category of failures, i.e., stochastic bad events defined by the polynomial-time predicate F (where F may be a different predicate for each protocol).

For simplicity, we shall call our $\mathcal{F}_{\text{mine}}$-hybrid protocol Π_{ideal} — for the three different protocols, Π_{ideal} is a different protocol; nonetheless, the same proofs hold for all three protocols.

E.3 Hybrid 1

Hybrid 1 is defined just like our earlier $\mathcal{F}_{\text{mine}}$-hybrid protocol but with the following modifications:

- $\mathcal{F}_{\text{mine}}$ chooses random PRF keys for all nodes at the very beginning, and let sk_i denote the PRF key chosen for the i-th node.
- Whenever a node i makes a $\texttt{mine}(\mathsf{m})$ query, $\mathcal{F}_{\text{mine}}$ determines the outcome of the coin flip as follows: compute $\rho \leftarrow \mathsf{PRF}_{\mathsf{sk}_i}(\mathsf{m})$ and use $\rho < D_p$ as the coin.
- Whenever $\mathcal{A}$ adaptively corrupts a node i, $\mathcal{F}_{\text{mine}}$ discloses sk_i to $\mathcal{A}$.

Lemma 21. *For any p.p.t. $(\mathcal{A}, \mathcal{Z})$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any κ, the bad events defined by F do not happen in Hybrid 1 with probability $1 - \mathsf{negl}(\kappa)$.*

Proof. Let f be the number of adaptive corruptions made by $\mathcal{A}$. To prove this lemma we must go through a sequence of inner hybrids over the number of adaptive corruptions made by the adversary $\mathcal{A}$.

Hybrid $1.f$. Hybrid $1.f$ is defined almost identically as Hybrid 1 except the following modifications: Suppose that $\mathcal{A}$ makes the last corruption query in round t and for node i. Whenever the ideal functionality $\mathcal{F}_{\text{mine}}$ in Hybrid 1 would have called $\mathsf{PRF}_{\mathsf{sk}_j}(\mathsf{m})$ for any j that is honest-forever and in some round $t' \geq t$, in Hybrid $1.f$, we replace this call with a random string.

Claim 4. *Suppose that the PRF scheme satisfies pseudorandomness under selective opening. Then, if for any p.p.t. $(\mathcal{A}, \mathcal{Z})$ and any κ, the bad events defined by F do not happen in Hybrid $1.f$ with probability at least $\mu(\kappa)$, then for any p.p.t. $(\mathcal{A}, \mathcal{Z})$ and κ, the bad events defined by F do not happen in Hybrid 1 with probability at least $\mu(\kappa) - \mathsf{negl}(\kappa)$.*

Proof. Suppose for the sake of contradiction that the claim does not hold. We can then construct a PRF adversary $\mathcal{A}'$ that breaks pseudorandomness under selective opening with non-negligible probability. $\mathcal{A}'$ plays $\mathcal{F}_{\text{mine}}$ when interacting with $\mathcal{A}$. $\mathcal{A}'$ is also interacting with a PRF challenger. In the beginning, for every node, $\mathcal{A}'$ asks the PRF challenger to create a PRF instance for that node. Whenever $\mathcal{F}_{\text{mine}}$ needs to evaluate a PRF, $\mathcal{A}'$ forwards the query to the PRF challenger. This continues until $\mathcal{A}$ makes the last corruption query, i.e., the f-th corruption query — suppose this last corruption query is made in round t and the node to corrupt is i. At this moment,

$\mathcal{A}'$ discloses sk_i to the adversary. However, whenever Hybrid 1 would have needed to compute $\mathsf{PRF}_{\mathsf{sk}_j}(\mathsf{m})$ for any j that is honest-forever and in some round $t' \geq t$, $\mathcal{A}'$ makes a challenge query to the PRF challenger for the j-th PRF instance and on the message queried. Notice that if the PRF challenger returned random answers to challenges, $\mathcal{A}$'s view in this interation would be identically distributed as Hybrid $1.f$. Otherwise, if the PRF challenger returned true answers to challenges, $\mathcal{A}$'s view in this interation would be identically distributed as Hybrid 1. □

Hybrid $1.f'$. Hybrid $1.f'$ is defined almost identically as Hybrid $1.f$ except the following modification: whenever $\mathcal{A}$ makes the last corruption query — suppose that this query is to corrupt node i and happens in round t — the ideal functionality $\mathcal{F}_{\text{mine}}$ does not disclose sk_i to $\mathcal{A}$.

Claim 5. *If for any p.p.t. $(\mathcal{A}, \mathcal{Z})$ and any κ, the bad events defined by F do not happen in Hybrid $1.f'$ with probability at least $\mu(\kappa)$, then for any p.p.t. $(\mathcal{A}, \mathcal{Z})$ and κ, the bad events defined by F do not happen in Hybrid $1.f$ with probability at least $\mu(\kappa)$.*

Proof. We observe the following: once the last corruption query is made in round t for node i, given that for any $t' \geq t$, any honest-forever node's coins are completely random. Thus whether or not the adversary receives the last corruption key does not help it to cause the relevant bad events to occur. Specifically in this case, at the moment the last corruption query is made — without loss of generality assume that the adversary makes all possible corrupt mining attempts — then whether the polynomial-checkable bad events defined by F take place is fully determined by $\mathcal{F}_{\text{mine}}$'s random coins and independent of any further actions of the adversary at this point. □

Hybrid $1.f''$. Hybrid $1.f''$ is defined almost identically as Hybrid $1.f'$ except the following modification: suppose that the last corruption query is to corrupt node i and happens in round t; whenever the ideal functionality $\mathcal{F}_{\text{mine}}$ in Hybrid $1.f'$ would have called $\mathsf{PRF}(\mathsf{sk}_i, \mathsf{m})$ in some round $t' \geq t$ (for the node i that is last corrupt), in Hybrid $1.f''$, we replace this call's outcome with a random string.

Claim 6. *Suppose that the PRF scheme satisfies pseudorandomness under selective opening. Then, if for any p.p.t. $(\mathcal{A}, \mathcal{Z})$ and any κ, the bad events defined by F do not happen in Hybrid $1.f''$ with probability at least $\mu(\kappa)$, then for any p.p.t. $(\mathcal{A}, \mathcal{Z})$ and κ, the bad events defined by F do not happen in Hybrid $1.f'$ with probability at least $\mu(\kappa) - \mathsf{negl}(\kappa)$.*

Proof. Suppose for the sake of contradiction that the claim does not hold. We can then construct a PRF adversary $\mathcal{A}'$ that breaks pseudorandomness under selective opening with non-negligible probability. $\mathcal{A}'$ plays the $\mathcal{F}_{\text{mine}}$ when interacting with $\mathcal{A}$. $\mathcal{A}'$ is also interacting with a PRF challenger. In the beginning, for every node, $\mathcal{A}'$ asks the PRF challenger to create a PRF instance for that node. Whenever $\mathcal{F}_{\text{mine}}$ needs to evaluate a PRF, $\mathcal{A}'$ forwards the query to the PRF challenger. This continues until $\mathcal{A}$ makes the last corruption query, i.e., the f-th corruption query — suppose this last corruption query is made in round t and the node to corrupt is i. At this moment, $\mathcal{A}'$ does not disclose sk_i to the adversary and does not query the PRF challenger to corrupt i's secret key either. Furthermore, whenever Hybrid $1.f'$ would have called $\mathsf{PRF}_{\mathsf{sk}_i}(\mathsf{m})$ in some round $t' \geq t$, $\mathcal{A}$ now calls the PRF challenger for the i-th PRF instance and on the specified challenge message, it uses the answer from the PRF challenger to replace the $\mathsf{PRF}_{\mathsf{sk}_i}(\mathsf{m})$ call. Notice that if the PRF challenger returned random answers to challenges, $\mathcal{A}$'s view in this interation would be identically distributed as Hybrid $1.f''$. Otherwise, if the PRF challenger returned true answers to challenges, $\mathcal{A}$'s view in this interation would be identically distributed as Hybrid $1.f'$. □

We can extend the same argument continuing with the following sequence of hybrids such that we can replace more and more PRF evaluations at the end with random coins, and withhold more and more PRF secret keys from $\mathcal{A}$ upon adaptive corruption queries — and nonetheless the probability that the security properties get broken will not be affected too much.

Hybrid $1.(f-1)$**.** Suppose that $\mathcal{A}$ makes the last but second corruption query for node i and in round t. Now, for any node j that is still honest in round t (not including node i), if $\mathsf{PRF}_{\mathsf{sk}_j}(\mathsf{m})$ is needed by the ideal functionality in some round $t' \geq t$, the PRF call's outcome will be replaced with random. Otherwise Hybrid $1.(f-1)$ is the same as Hybrid $1.f''$.

Claim 7. *Suppose that the PRF scheme satisfies pseudorandomness under selective opening. Then, if for any p.p.t.* $(\mathcal{A}, \mathcal{Z})$ *and any* κ*, the bad events defined by* F *do not happen in Hybrid* $1.(f-1)$ *with probability at least* $\mu(\kappa)$*, then for any p.p.t.* $(\mathcal{A}, \mathcal{Z})$ *and* κ*, the bad events defined by* F *do not happen in Hybrid* $1.f''$ *with probability at least* $\mu(\kappa) - \mathsf{negl}(\kappa)$*.*

Proof. Similar to the reduction between the $\mathcal{F}_{\text{mine}}$-hybrid protocol and Hybrid $1.f$. □

Hybrid $1.(f-1)'$**.** Almost the same as Hybrid $1.(f-1)$, but without disclosing the secret key to $\mathcal{A}$ upon the last but second corruption query.

Claim 8. *If for any p.p.t.* $(\mathcal{A}, \mathcal{Z})$ *and any* κ*, the bad events defined by* F *do not happen in Hybrid* $1.(f-1)'$ *with probability at least* $1-\mu(\kappa)$*, then for any p.p.t.* $(\mathcal{A}, \mathcal{Z})$ *and* κ*, the bad events defined by* F *do not happen in Hybrid* $1.(f-1)$ *with probability at least* $1-\mu(\kappa)$*.*

Proof. The proof is similar to the reduction between Hybrid $1.f$ and Hybrid $1.f'$, but with one more subtlety: in Hybrid $1.(f-1)$, upon making the last but second adaptive corruption query for node i in round t, for any $t' \geq t$ and any node honest in round t (not including i but including the last node to corrupt), all coins are random. Due to this, we observe that if there is a p.p.t. adversary $\mathcal{A}$ that can cause the bad events defined by F to occur with probability μ for Hybrid $1.(f-1)$, then there is another p.p.t. adversary $\mathcal{A}'$ such that upon making the last but second corruption query, it would immediately make the last corruption query in the same round as t corrupting an arbitrary node (say, the one with the smallest index and is not corrupt yet), and $\mathcal{A}'$ can cause the bad events defined by F to occur with probability at least μ in Hybrid $1.(f-1)$.

Now, we argue that if such an $\mathcal{A}'$ can cause the bad events defined by F to occur in Hybrid $1.(f-1)$ with probability μ, there must be an adversary $\mathcal{A}''$ that can cause the bad events defined by F to occur in Hybrid $1.(f-1)'$ with probability μ too. In particular, $\mathcal{A}''$ will simply run $\mathcal{A}'$ until $\mathcal{A}'$ makes the last but second corruption query. At this point $\mathcal{A}''$ makes an additional corruption query for an arbitrary node that is not yet corrupt. At this point, clearly whether bad events defined by F would occur is independent of any further action of the adversary — and although in Hybrid $1.(f-1)'$, $\mathcal{A}''$ does not get to see the secret key corresponding to the last but second query, it still has the same probability of causing the relevant bad events to occur as the adversary $\mathcal{A}'$ in Hybrid $1.(f-1)$. □

Hybrid $1.(f-1)''$**.** Suppose that $\mathcal{A}$ makes the last but second corruption query for node i and in round t. Now, for any node j that is still honest in round t as well as node $j = i$, if the ideal functionality needs to call $\mathsf{PRF}_{\mathsf{sk}_j}(\mathsf{m})$ in some round $t' \geq t$ the PRF's outcome will be replaced with random. Otherwise Hybrid $1.(f-1)''$ is identical to $1.(f-1)'$.

Due to the same argument as that of Claim 6, we may conclude that if for any p.p.t. $(\mathcal{A}, \mathcal{Z})$ and any κ, the bad events defined by F do not happen in Hybrid $1.(f-1)''$ with probability at

least $\mu(\kappa)$, then for any p.p.t. $(\mathcal{A}, \mathcal{Z})$ and κ, the bad events defined by F do not happen in Hybrid $1.(f-1)'$ with probability at least $\mu(\kappa) - \mathsf{negl}(\kappa)$.

In this manner, we define a sequence of hybrids till in the end, we reach the following hybrid:

Hybrid 1.0**.** All PRFs evaluations in Hybrid 1 are replaced with random, and no secret keys are disclosed to $\mathcal{A}$ upon any adaptive corruption query.

It is not difficult to see that Hybrid 1.0 is identically distributed as the $\mathcal{F}_{\text{mine}}$-hybrid protocol. We thus conclude the proof of Lemma 21.

□

E.4 Hybrid 2

Hybrid 2 is defined almost identically as Hybrid 1, except that now the following occurs:

- Upfront, $\mathcal{F}_{\text{mine}}$ generates an honest CRS for the commitment scheme and the NIZK scheme and discloses the CRS to $\mathcal{A}$.
- Upfront, $\mathcal{F}_{\text{mine}}$ not only chooses secret keys for all nodes, but commits to the secret keys of these nodes, and reveals the commitments to $\mathcal{A}$.
- Every time $\mathcal{F}_{\text{mine}}$ receives a `mine` query from a so-far-honest node i and for the message m, it evaluates $\rho \leftarrow \mathsf{PRF}_{\mathsf{sk}_i}(\mathsf{m})$ and compute a NIZK proof denoted π to vouch for ρ. Now, $\mathcal{F}_{\text{mine}}$ returns ρ and π to $\mathcal{A}$.
- Whenever a node i becomes corrupt, $\mathcal{F}_{\text{mine}}$ reveals all secret randomness node i has used in commitments and NIZKs so far to $\mathcal{A}$ in addition to revealing its PRF secret key sk_i.

Lemma 22. *Suppose that the commitment scheme is computationally adaptive hiding under selective opening, and the NIZK scheme is non-erasure computational zero-knowledge, Then, for any p.p.t.* $(\mathcal{A}, \mathcal{Z})$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for any* κ*, the bad events defined by* F *do not happen in Hybrid 2 with probability* $1 - \mathsf{negl}(\kappa)$*.*

Proof. The proof is standard and proceeds in the following internal hybrid steps.

- **Hybrid 2.A.** Hybrid 2.A is the same as Hybrid 2 but with the following modifications. $\mathcal{F}_{\text{mine}}$ calls simulated NIZK key generation instead of the real one, and for nodes that remain honest so-far, $\mathcal{F}_{\text{mine}}$ simulate their NIZK proofs without needing the nodes' PRF secret keys. Whenever an honest node i becomes corrupt, $\mathcal{F}_{\text{mine}}$ explains node i's simulated NIZKs using node i's real sk_i and randomness used in its commitment, and supplies the explanations to $\mathcal{A}$.

  Claim 9. *Hybrid 2.A and Hybrid 2 are computationally indistinguishable from the* view *of* $\mathcal{Z}$*.*

  Proof. Straightforward due to the non-erasure computational zero-knowledge property of the NIZK. □

- **Hybrid 2.B.** Hybrid 2.B is almost identical to Hybrid 2.A but with the following modifications. $\mathcal{F}_{\text{mine}}$ calls the simulated CRS generation for the commitment scheme. When generating public keys for nodes, it computes simulated commitments without using the nodes' real sk_i's. When a node i becomes corrupt, it will use the real sk_i to compute an explanation for the earlier simulated commitment. Now this explanation is supplied to the NIZK's explain algorithm to explain the NIZK too.

Claim 10. *Hybrid 2.A and Hybrid 2.B are computationally indistinguishable from the view of the environment $\mathcal{Z}$.*

Proof. Straightforward by the "computational hiding under selective opening" property of the commitment scheme. □

Claim 11. *If for any p.p.t. $(\mathcal{A}, \mathcal{Z})$ and any κ, the bad events defined by F do not happen in Hybrid 1 with probability at least $\mu(\kappa)$, then for any p.p.t. $(\mathcal{A}, \mathcal{Z})$ and κ, then the bad events defined by F do not happen in Hybrid 2.B with probability at least $\mu(\kappa)$.*

Proof. Given an adversary $\mathcal{A}$ that attacks Hybrid 2.B, we can construct an adversary $\mathcal{A}'$ that attacks Hybrid 1. $\mathcal{A}'$ will run $\mathcal{A}$ internally. $\mathcal{A}'$ runs the simulated CRS generations algorithms for the commitment and NIZK, and sends the simulated CRSes to $\mathcal{A}$. It then runs the simulated commitment scheme and sends simulated commitments to $\mathcal{A}$ (of randomly chosen sk_i for every i). Whenever $\mathcal{A}$ tries to mine a message, $\mathcal{A}'$ can intercept this mining request, forward it to its own $\mathcal{F}_{\text{mine}}$. If successful, $\mathcal{A}'$ can sample a random number $\rho > D_p$; else it samples a random number $\rho \leq D_p$. It then calls the simulated NIZK prover using ρ to simulate a NIZK proof and sends it to $\mathcal{A}$. Whenever $\mathcal{A}$ wants to corrupt a node i, $\mathcal{A}'$ corrupts it with its $\mathcal{F}_{\text{mine}}$, obtains sk_i, and then runs the $\mathsf{Explain}$ algorithms of the commitment and NIZK schemes and discloses the explanations to $\mathcal{A}$. Clearly $\mathcal{A}$'s view in this protocol is identically distributed as in Hybrid 2.B. Moreover, if $\mathcal{A}$ succeedings in causing the bad events defined by F to happen, clearly $\mathcal{A}'$ will too. □

□

E.5 Hybrid 3

Hybrid 3 is almost identical as Hybrid 2 except with the following modifications. Whenever an already corrupt node makes a mining query to $\mathcal{F}_{\text{mine}}$, it must supply a ρ and a NIZK proof π. $\mathcal{F}_{\text{mine}}$ then verifies the NIZK proof π, and if verification passes, it uses $\rho < D_p$ as the result of the coin flip.

Lemma 23. *Assume that the commitment scheme is perfectly binding, and the NIZK scheme satisfies perfect knowledge extraction. Then, for any p.p.t. $(\mathcal{A}, \mathcal{Z})$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any κ, the bad events defined by F do not happen in Hybrid 3 except with probability $\mathsf{negl}(\kappa)$.*

Proof. We can replace the NIZK's CRS generation Gen with Gen_1 which generates a CRS that is identically distributed as the honest Gen, but additionally generates an extraction trapdoor denoted τ_1. Now, upon receiving $\mathcal{A}$'s NIZK proof π, $\mathcal{F}_{\text{mine}}$ performs extraction. The lemma follows by observing that due to the perfect knowledge extraction of the NIZK and the perfect binding property of the commitment scheme, it holds except with negligible probability that the extracted witness does not match the node's PRF secret key that $\mathcal{F}_{\text{mine}}$ had chosen upfront. □

In the lemma below, when we say that "*assume that the cryptographic building blocks employed are secure*", we formally mean that the pseudorandom function family employed is secure; the non-interactive zero-knowledge proof system that satisfies non-erasure computational zero-knowledge and perfect knowledge extraction; the commitment scheme is computationally hiding under selective opening and perfectly binding; and for the synchronous honest majority protocol, additionally assume that the signature scheme is secure.

Lemma 24. *Assume the cryptographic building blocks employed are secure. Then, for any p.p.t.* $(\mathcal{A}, \mathcal{Z})$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for any* $\kappa \in \mathbb{N}$*, relevant security properties (including consistency, validity, and termination) are preseved with all but* $\mathsf{negl}(\kappa)$ *probability in Hybrid 3.*

Proof. As mentioned, only two types of bad events can possibly lead to breach of the relevant security properties: 1) signature failure; and 2) bad events defined by F. Thus the lemma follows in a straightforward fashion by taking a union bound over the two. □

E.6 Real-World Execution

We now show that the real-world protocol is just as secure as Hybrid 3 — recall that the security properties we care about include consistency, validity, and termination.

Lemma 25. *If there is some p.p.t.* $(\mathcal{A}, \mathcal{Z})$ *that causes the relevant security properties to be broken in the real world with probability* μ*, then there is some p.p.t.* $\mathcal{A}'$ *such that* $(\mathcal{A}', \mathcal{Z})$ *can cause the relevant security properties to be broken in Hybrid 3 with probability at least* μ*.*

Proof. We construct the following $\mathcal{A}'$:

- $\mathcal{A}'$ obtains CRSes for the NIZK and the commitment scheme from its $\mathcal{F}_{\text{mine}}$ and forwards them to $\mathcal{A}$. $\mathcal{A}'$ also forwards the PKI it learns from $\mathcal{F}_{\text{mine}}$ to $\mathcal{A}$.
- Whenever $\mathcal{A}$ corrupts some node, $\mathcal{A}'$ does the same with its $\mathcal{F}_{\text{mine}}$, and forwards whatever learned to $\mathcal{A}$.
- Whenever $\mathcal{A}$ sends some message to an honest node, for any portion of the message that is a "mined message" of any type, let (m, ρ, π) denote this mined message — we assume that m contains the purported miner of this message denoted i.
  - $\mathcal{A}'$ checks the validity of π and that $\rho < D_p$ for an appropriate choice of p depending on the message's type; ignore the message if the checks fail;
  - if the purported sender i is an honest node and node i has not successfully mined m with $\mathcal{F}_{\text{mine}}$, record a $\mathsf{forgery}$ event and simply ignore this message. Otherwise, continue with the following steps.
  - if the purported sender i is a corrupt node: $\mathcal{A}'$ issues a corresponding mining attempt to $\mathcal{F}_{\text{mine}}$ on behalf of i with the corresponding ρ and π if no such mining attempt has been made before;
  - Finally, $\mathcal{A}'$ forwards m to the destined honest on behalf of the corrupt sender.
- Whenever $\mathcal{A}'$ receives some message from an honest node (of Hybrid 3): for every portion of the message that is a "mined message" of any type, at this point $\mathcal{A}'$ must have heard from $\mathcal{F}_{\text{mine}}$ the corresponding ρ, and π terms. $\mathcal{A}'$ augments the message with these terms and forwards the resulting message to $\mathcal{A}$.

Note that conditioned on views (determined by all randomness of the execution) with no $\mathsf{forgery}$ event then either the relevant bad events occur both in Hybrid 2 and the real-world execution, or occur in neither. For views with $\mathsf{forgery}$ events, it is not difficult to see that if Hybrid 2 (on this view) does not incur the relevant bad events, then neither would the real-world execution (for this view). □

F Extra Warmup: Selective Opening Security for PRFs

Theorem 10 (Restatement of Theorem 9)**.** *Any secure PRF family satisfies pseudorandomness under selective opening by Definition 3 (with polynomial loss in the security reduction).*

Proof. **Single-selective-challenge selective opening security.** In the single-selective challenge version of the game, the adversary commits to a challenge identifier i^* upfront during the security game, such that later, challenge queries can only be made for the committed index i^*.

First, we can show that any secure PRF family would satisfy single-selective-challenge selective opening security. Suppose that there is an efficient adversary $\mathcal{A}$ that can break the single-selective-challenge selective opening security game for some PRF family. We construct a reduction $\mathcal{R}$ that leverages $\mathcal{A}$ to break the PRF's security. The reduction $\mathcal{R}$ interacts with a PRF challenger as well as $\mathcal{A}$. $\mathcal{R}$ generates PRF keys for all instances other than i^* and answers non-i^* evaluation and corruption queries honestly. For i^*, $\mathcal{A}$'s evaluation requests are forwarded to the PRF challenger.

We consider the following three hybrids:

1. The PRF challenger has a real, randomly sampled PRF function from the corresponding family, and $\mathcal{R}$ answers $\mathcal{A}$'s challenge queries on i^* with random answers;
2. The PRF challenger has a random function, and $\mathcal{R}$ answers $\mathcal{A}$'s challenge queries on i^* by forwarding the PRF challenger's answers (or equivalently by relying with random answers); and
3. The PRF challenger has a real, randomly sampled PRF function from the corresponding family, and $\mathcal{R}$ answers $\mathcal{A}$'s challenge queries on i^* by forwarding the PRF challenger's answers.

It is not difficult to see that $\mathcal{A}$'s view in hybrid 1 is identical to its view in the single-selective challenge selective opening security game when $b = 0$; its view in hybrid 3 is identical to its view in the single-selective challenge selective opening security game when $b = 1$. Due to the security of the PRF, it is not difficult to see that any adjacent pair of hybrids are indistinguishable.

Single-challenge selective opening security. In the single-challenge selective opening version of the game, the adversary can only make challenge queries for a single i^* but it need not commit to i^* upfront at the beginning of the security game.

We now argue that any PRF that satisfies single-selective-challenge selective opening security must satisfy single-challenge selective opening security with a polynomial security loss. The proof of this is straightforward. Suppose that there is an efficient adversary $\mathcal{A}$ that can break the single-challenge selective opening security of some PRF family, we can then construct an efficient reduction $\mathcal{R}$ that breaks the single-selective-challenge selective opening security of the PRF family. Basically the reduction $\mathcal{R}$ guesses at random upfront which index i^* the adversary $\mathcal{A}$ will choose for challenge queries. $\mathcal{R}$ then forwards all of $\mathcal{A}$'s queries to the challenger of the single-selective-challenge selective opening security security game. If the guess later turns out to be wrong, the reduction simply aborts and outputs a random guess b'. Otherwise, it outputs the same output as $\mathcal{A}$. Suppose that $\mathcal{A}$ creates q instances of PRFs then we can conclude that $\mathcal{R}$ guesses correctly with probability at least $1/q$. Thus whatever advantage $\mathcal{A}$ has in breaking the single-challenge selective opening security, $\mathcal{R}$ has an advantage that is $1/q$ fraction of $\mathcal{A}$'s advantage in breaking the single-selective-challenge selective opening security of the PRF family.

Selective opening security. Finally, we show that any PRF family that satisfies single-challenge selective opening security must also satisfy selective opening security (i.e., Definition 3) with a

polynomial security loss. This proof can be completed through a standard hybrid argument in which we replace the challenge queries from real to random one index at a time (where replacement is performed for all queries of the i-th new index that appeared in some challenge query). $\square$