Noname manuscript No.
(will be inserted by the editor)

Communication Complexity of Byzantine Agreement, Revisited

Ittai Abraham · T-H. Hubert Chan · Danny Dolev · Kartik Nayak · Rafael Pass · Ling Ren · Elaine Shi

Received: / Accepted:

Abstract As Byzantine Agreement (BA) protocols find application in large-scale decentralized cryptocurrencies, an increasingly important problem is to design BA protocols with improved communication complexity. A few existing works have shown how to achieve subquadratic BA under an *adaptive* adversary. Intriguingly, they all make a common relaxation about the adaptivity of the attacker, that is, if an honest node sends a message and then gets corrupted in some round, the adversary *cannot erase the message that was already sent* — henceforth we say that such an adversary cannot perform "after-the-fact removal". By contrast, many (super-)quadratic BA protocols in the literature can tolerate after-the-fact removal. In this paper, we first prove that disallowing after-the-fact removal is necessary for achieving subquadratic-communication BA.

This work is partially supported by The Federmann Cyber Security Center in conjunction with the Israel National Cyber Directorate. T-H. Hubert Chan was partially supported by the Hong Kong RGC under the grant 17200418.

VMware Research
E-mail: iabraham@vmware.com

The University of Hong Kong
E-mail: hubert@cs.hku.hk

The Hebrew University of Jerusalem
E-mail: dolev@cs.huji.ac.il

Duke University
E-mail: kartik@cs.duke.edu

Cornell Tech
E-mail: rafael@cs.cornell.edu

University of Illinois, Urbana-Champaign
E-mail: renling@illinois.edu

Cornell University
E-mail: runting@gmail.com

Next, we show new subquadratic binary BA constructions (of course, assuming no after-the-fact removal) that achieve near-optimal resilience and expected constant rounds under standard cryptographic assumptions and a public-key infrastructure (PKI) in both synchronous and partially synchronous settings. In comparison, all known subquadratic protocols make additional strong assumptions such as random oracles or the ability of honest nodes to erase secrets from memory, and even with these strong assumptions, no prior work can achieve the above properties. Lastly, we show that some setup assumption is necessary for achieving subquadratic multicast-based BA.

Keywords Byzantine agreement · communication complexity · subquadratic · lower bounds

1 Introduction

Byzantine agreement (BA) [33] is a central abstraction in distributed systems. Classic BA protocols [9, 17, 19] require all players to send messages to all other players, and thus, n-player BA requires at least n^2 communication complexity. Such protocols are thus not well suited for *large-scale distributed systems* such as decentralized cryptocurrencies [35]. A fundamental problem is to design BA protocols with improved communication complexity.

In fact, in a model with *static* corruption, this is relatively easy. For example, suppose there are at most $f < (\frac{1}{2} - \epsilon)n$ corrupt players where ϵ is a positive constant; further, assume there is a trusted common random string (CRS) that is chosen independently of the adversary's (static) corruption choices. Then, we can use the CRS to select a λ-sized committee of players. Various elegant works have investigated how to weaken or remove the trusted set-up assumptions required for committee election and retain subquadratic communication [10,31]. Once a committee is selected, we can run any BA protocol among the committee, and let the committee members send their outputs to all "non-committee" players who could then output the majority bit. This protocol works as long as there is an honest majority on the committee. Thus, the error probability is bounded by $\exp(-\Omega(\lambda))$ due to a standard Chernoff bound.

Such a committee-based approach, however, fails if we consider an *adaptive attacker*. Such an attacker can simply observe which players are on the committee, then corrupt them, and thereby control the entire committee! A natural and long-standing open question is thus whether subquadratic communication is possible w.r.t. an adaptive attacker:

> *Does there exist a BA protocol with subquadratic communication complexity that resists adaptive corruption of players?*

This question has been partially answered in a few prior works. First, a breakthrough work by King and Saia [30] presented a BA protocol with communication complexity $O(n^{1.5})$. More recent works studied practical constructions motivated by cryptocurrency applications: notably the celebrated Nakamoto

consensus [24, 35] can reach agreement in subquadratic communication assuming idealized proof-of-work. Subsequently, several so-called "proof-of-stake" constructions [12, 15] also showed how to realize BA with subquadratic communication. All of the above works assume synchrony and all of them tolerate adaptive corruptions.

What is both intriguing and unsatisfying is that all these works happen to make a common relaxing assumption about the adaptivity of the adversary, namely, if adversary adaptively corrupts an honest player i who has just sent a message m in round r, the adversary is unable to erase the honest message m sent in round r. Henceforth we say that such an adversary is incapable of *after-the-fact removal*. In comparison, many natural $\Omega(n^2)$-communication BA protocols [1, 17, 29] can be proven secure even if the adversary is capable of after-the-fact removal – henceforth referred to as a *strongly adaptive* adversary. That is, if an honest player i sends a message m in round r, a strongly adaptive adversary (e.g., who controls the egress routers of many players) can observe m and then decide to corrupt i and erase the message m that player i has just sent in round r. This mismatch in model naturally raises the following question:

> Is disallowing after-the-fact removal necessary for achieving subquadratic-communication BA?

Main result 1: disallowing "after-the-fact" removal is necessary. Our first contribution is a new lower bound showing that any (possibly randomized) BA protocol must incur at least $\Omega(f^2)$ messages in expectation in the presence of a strongly adaptive adversary where f denotes the number of corrupt players. This quadratic lower bound on the number of messages implies a quadratic lower bound on the communication complexity in terms of bits. The proof of our lower bound is inspired by the work of Dolev and Reischuk [16], who showed that any *deterministic* BA protocol must incur $\Omega(f^2)$ messages even against a *static* adversary. We remark our lower bound (as well as Dolev-Reischuk) holds in a very strong sense: even when making common (possibly very strong) assumptions such as proof-of-work and random oracles, and even under a more constrained *omission* adversary who is only allowed to omit messages sent from and to corrupt players, but does not deviate from the protocol otherwise.

Theorem 1 (Impossibility of BA with subquadratic communication w.r.t. a strongly adaptive adversary) *Any (possibly randomized) BA protocol must in expectation incur at least $\Omega(f^2)$ communication in the presence of a strongly adaptive adversary capable of performing after-the-fact removal, where f denotes the number of corrupt players.*

Main result 2: near-optimal subquadratic BA with minimal assumptions. On the upper bound front, we present subquadratic BA protocols that, besides the necessary "no after-the-fact removal" assumption, rely only on standard cryptographic and setup assumptions. Our protocols achieve near-optimal resilience and expected constant rounds.

Our results improve upon existing works in two major aspects. Firstly, besides "no after-the-fact removal", all existing subquadratic protocols make very strong *additional* assumptions, such as random oracles [12, 15] or proof-of-work [35]. In particular, some works [12, 30] assume the ability of honest players to securely erase secrets from memory and that adaptive corruption cannot take place between when an honest player sends a message and when it erases secrets from memory. Such a model is referred to as the "erasure model" in the cryptography literature and as "ephemeral keys" in Chen and Micali [12]. To avoid confusing the term with "after-the-fact message removal", we rename it the *memory-erasure model* in this paper. Secondly, and more importantly, even with those strong assumptions, existing protocols do not achieve the above properties (cf. Section 1.1).

The multicast model. In a large-scale peer-to-peer network, it is usually much cheaper for a player to multicast the same message to everyone, than to unicast n different messages (of the same length) to n different players — even though the two have identical communication complexity in the standard pairwise model. Indeed, all known consensus protocols deployed in a decentralized environment (e.g. Bitcoin, Ethereum) work in the multicast fashion. Since our protocols are motivated by these large-scale peer-to-peer networks, we design our protocols to be multicast-based.

A multicast-based protocol is said to have *multicast complexity* C if the total number of bits multicast by all honest players is upper bounded by C. Clearly, a protocol with multicast complexity C has communication complexity nC. Thus, to achieve subquadratic communication complexity, we need to design a protocol in which only a sublinear (in n) number of players multicast.

Theorem 2 *Assuming standard cryptographic assumptions and a public-key infrastructure (PKI),*

1. *For any constant $0 < \epsilon < 1/2$, there exists a synchronous BA protocol with expected $O(\chi \cdot \mathsf{poly}\log(\kappa))$ multicast complexity, expected $O(1)$ round complexity, and $\mathsf{negl}(\kappa)$ error probability that tolerates $f < (1-\epsilon)n/2$ adaptively corrupted players out of n players in total.*
2. *For any constant $0 < \epsilon < 1/3$, there exists a partially synchronous BA protocol with expected $O(\chi \cdot \mathsf{poly}\log(\kappa))$ multicast complexity, expected $O(\Delta \cdot \mathsf{poly}\log(\kappa))$ time, and $\mathsf{negl}(\kappa)$ error probability that tolerates $f < (1-\epsilon)n/3$ adaptively corrupted players out of n players in total.*

In both statements, κ is a security parameter, $\mathsf{negl}(\cdot)$ is a negligible function, and χ is a computational security parameter; $\chi = \mathsf{poly}(\kappa)$ under standard cryptographic assumptions and $\chi = \mathsf{poly}\log(\kappa)$ if we assume sub-exponential security of the cryptographic primitives employed; In the second statement, Δ is an unknown upper bound on the message delay.

Our construction requires a random verifiable function (VRF) that is secure against an adaptive adversary. Here, adaptive security means security under *selective opening* of corrupt players' secret keys, which is a different notion

of adaptivity from in some prior works [5, 25]. Most previously known VRF constructions [5, 25, 34] do not provide security under an adaptive adversary. Chen and Micali [12] use random oracles (RO) and unique signatures to construct an adaptively secure VRF.

Main result 3: on the necessity of setup assumptions. In light of the above Theorem 2, we additionally investigate whether the remaining setup PKI assumption is necessary. We show that if one insists on a multicast-based protocol, indeed some form of setup assumption is necessary for achieving sublinear multicast complexity. Specifically, we show that without any setup assumption, i.e., under the plain authenticated channels model, a (possibly randomized) protocol that solves BA with C multicast complexity with probability $p > 5/6$ can tolerate no more than C adaptive corruptions.

Theorem 3 (Impossibility of sublinear multicast BA without setup assumptions) *In a plain authenticated channel model without setup assumptions, no protocol can solve BA using C multicast complexity with probability $p > 5/6$ under C adaptive corruptions.*

We remark that this lower bound also applies more generally to protocols in which few players (i.e., less than C nodes) speak (multicast-style protocols are a special case). Also note that there exist protocols with subquadratic communication and no setup assumptions that rely on many players to speak [30].

Organization. The rest of the paper is organized as follows. Section 1.1 reviews related work. Section 2 presents the model and definitions of BA with additional details in Section 9. Section 3 proves Theorem 1. Sections 4, 5, and 6 construct adaptively secure BA protocols to prove Theorem 2. Section 7 proves Theorem 3.

1.1 Related Work

Dolev and Reischuk [16] proved that quadratic communication is necessary for any deterministic BA protocol. Inspired by their work, we show a similar communication complexity lower bound for randomized protocols, but now additionally assuming that the adversary is strongly adaptive.

A number of works explored randomized BA protocols [3, 36] to achieve expected constant round complexity [1, 20, 29] even under a strongly adaptive adversary. A line of works [13, 23, 28] focused on a simulation-based stronger notion of adaptive security for Byzantine Broadcast. These works have at least quadratic communication complexity.

King and Saia first observed that BA can be solved with subquadratic communication complexity if a small probability of error is allowed [30]. More recently Nakamoto-style protocols, based on either proof-of-work [35] or proof-of-stake [12, 15] also showed how to realize BA with subquadratic communication. Compared to our synchronous protocol in Section 5, these existing works on synchronous BA make other strong assumptions, and even with those

strong assumptions, cannot simultaneously achieve near-optimal resilience and expected constant rounds. Nakamoto consensus [35] assumes idealized proofs-of-work. Proof-of-stake protocols assume random oracles [12,15]. King-Saia [30] and Chen-Micali [12] assume memory-erasure. Nakamoto-style protocols [15,35] and King-Saia [30] cannot achieve expected constant rounds. Chen-Micali [12] have sub-optimal tolerance of $f < (\frac{1}{3} - \epsilon)n$.

Subsequent to our work, there have been several works presenting consensus protocols with efficient communication complexity. Chen et al. [11] presented a partially synchronous protocol and mention that this can be extended to obtain sub-quadratic communication in a manner similar to Chen-Micali [12] assuming erasures. Cohen et al. [14] and Blum et al. [6] independently present an *asynchronous* BA protocol with sub-quadratic communication. Cohen et el. assume a specific notion of a delayed adaptive adversary where an adversary, once it takes over a process, it cannot frontrun messages that the process had already sent when it was non-faulty. On the other hand, Blum et al. assume stronger computational assumptions and a trusted dealer. Tsimos et al. [37] present the first Byzantine Broadcast protocol with subcubic communication in the dishonest majority setting. In comparison, our work only considers the synchronous and partially synchronous setting tolerating up to one-half and up to one-third corruption respectively under an adaptive adversary. Dryja et al. [18] argue that if the value of Δ can be arbitrary, then no BA protocol with $o(n)$ multicasts in the partially synchronous model can tolerate an adaptive adversary. In comparison, adopting the partial synchrony model from CKPS [7], we assume that Δ is polynomically bounded and show protocol with $O(\lambda^2 \log \Delta)$ multicasts in expectation. Bhangale et al. [4] extend our work to obtain a BA protocol with $O(nl + n \cdot \mathsf{poly}(\kappa))$ communication complexity for l-bit long messages (κ is a security parameter).

2 Model and Definition

In this section, we present the model and definition. We provide some additional details in Section 9.

Communication model. We assume two different communication models for two different protocols. In Sections 4 and 5, we assume that the network is synchronous and the protocol proceeds in rounds. Every message sent by an honest node is guaranteed to be received by an honest recipient at the beginning of the next round.

In Section 6, we assume that the network is partially synchronous. There are multiple ways to define partial synchrony [19]. In this paper, we consider the unknown Δ variant, i.e., there exists a fixed message delay bound of Δ rounds but Δ is not known to any honest node.

We measure communication complexity by the number of bits sent by honest nodes. Our protocols in Sections 4, 5, and 6 use multicasts only, that is, whenever an honest node sends a message, it sends that message to all

nodes including itself. We say a protocol has multicast complexity C if the total number of bits multicast by honest nodes is bounded by C.

For ease of presentation, in the main body of the paper, we present protocols assuming an ideal functionality of VRF; in particular, we will measure communication complexity in terms of the number of messages and state results using a statistical security parameter λ. In the appendix, we show how to instantiate an adaptively secure VRF under standard cryptographic assumptions such as bilinear groups, and restate our results using κ, χ, and communication complexity in bits.

Adversary model. We assume a standard protocol execution model with n players (also called *nodes*) numbered $1, 2, \ldots, n$. We assume a trusted public-key infrastructure (PKI), so every node knows the public key of every other node. The adversary is denoted $\mathcal{A}$ and is polynomially bounded. $\mathcal{A}$ can *adaptively* corrupt nodes any time during the protocol execution after the trusted setup. Since a node can start as honest but become corrupt in the middle of the protocol, we will use the following terminology to improve clarity. At any time in the protocol, nodes that remain honest so far are referred to as *so-far-honest* nodes and nodes that remain honest till the end of the execution are referred to as *forever-honest* nodes. Nodes that are corrupt by the end of the execution are referred to as *eventually-corrupt* nodes (even before they are corrupted). The total number of eventually-corrupt nodes is at most f. 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 in each round once they become corrupt. We assume that when a so-far-honest node i multicasts a message m, it can immediately become corrupt in the same round and be made to send one or more messages in the same round. We prove our lower bound in Section 3 under a strongly adaptive adversary that can perform an after-the-fact removal, i.e, it can retract messages that have already been multicast before the node becomes corrupt. For our upper bounds in subsequent sections, we assume an adaptive adversary cannot perform such a retraction.

Agreement vs. broadcast. (Binary) Byzantine Agreement is typically studied in two forms. In the *broadcast version*, also called *Byzantine broadcast*, there is a *designated sender* (or simply *sender*) known to all nodes. Prior to protocol start, the sender receives an input $b \in \{0, 1\}$. A protocol solves Byzantine broadcast with probability p if it achieves the following properties with probability at least p.

- *Termination.* Every forever-honest node i outputs a bit b'_i.
- *Consistency.* If two forever-honest nodes output b'_i and b'_j respectively, then $b'_i = b'_j$.
- *Validity.* If the sender is forever-honest and the sender's input is b, then all forever-honest nodes output b.

In the *agreement version*, sometimes referred to as Byzantine "consensus" in the literature, there is no designated sender. Instead, each node i receives

an input bit $b_i \in \{0,1\}$. A protocol solves Byzantine agreement (BA) with probability p if it achieves the following properties with probability at least p.

- *Termination* and *Consistency* same as Byzantine broadcast.
- *Validity.* If all forever-honest nodes receive the same input bit b, then all forever-honest nodes output b.

With synchrony and PKI, the *agreement* version (where everyone receives input) can tolerate up to minority corruption [22] while the broadcast version can tolerate up to $n-1$ corruptions [17,33]. Under minority-corruption, the two versions are equivalent from a feasibility perspective, i.e., we can construct one from the other. Moreover, one direction of the reduction preserves communication complexity. Specifically, given an adaptively secure BA protocol (agreement version), one can construct an adaptively secure Byzantine Broadcast protocol by first having the designated sender multicasting its input to everyone, and then having everyone invoke the BA protocol. This way, if the BA protocol has subquadratic communication complexity (resp. sublinear multicast complexity), so does the resulting Byzantine Broadcast protocol. For this reason, we state all our upper bounds for BA and state all our lower bounds for Byzantine Broadcast — this makes both our upper- and lower-bounds stronger.

3 Communication Lower Bound Under a Strongly Adaptive Adversary

In this section, we prove that any (possibly randomized) BA protocol must in expectation incur at least $\Omega(f^2)$ communication in the presence of a strongly adaptive adversary capable of performing after-the-fact removal. For the reasons mentioned in Section 2, we prove our lower bound for Byzantine Broadcast (which immediately applies to BA). Our proof strategy builds on the classic Dolev-Reischuk lower bound [16, Theorem 2], which shows that in every deterministic Byzantine Broadcast protocol honest nodes need to send at least $\Omega(f^2)$ messages.

Warmup: the Dolev-Reischuk lower bound. We first explain the Dolev-Reischuk proof at a high level. For an asymptotic lower bound, we can focus on the case of an even f. Observe that for a deterministic protocol, an execution is completely determined by the input (of the designated sender) and the adversary's strategy. Consider the following adversary $\mathcal{A}$: $\mathcal{A}$ corrupts a set V of $f/2$ nodes that does not include the designated sender. Let U denote the set of remaining nodes. All nodes in V behave like honest nodes, except that (i) they ignore the first $f/2$ messages sent to them, and (ii) they do not send messages to each other. Suppose the honest designated sender has input 0. For validity to hold, all honest nodes must output 0.

If at most $(f/2)^2$ messages are sent to V in the above execution, then there exists a node $p \in V$ that receives at most $f/2$ messages. Now, define another adversary $\mathcal{A}'$ almost identically as $\mathcal{A}$ except that: (i) $\mathcal{A}'$ does not corrupt p, (ii) $\mathcal{A}'$ corrupts all nodes in U that send p messages (possibly including the

designated sender), prevents them from sending any messages to p, but behaves honestly to other nodes. Since p receives at most $f/2$ messages under $\mathcal{A}$, $\mathcal{A}'$ corrupts at most f nodes.

Observe that honest nodes in U receive identical messages from all other nodes in the two executions. So these nodes still output 0 under $\mathcal{A}'$. However, p does not receive any message but has to output some value. If this value is 1, consistency is violated. If p outputs 0 when receiving no messages, we can let the sender send 1 under $\mathcal{A}$ and derive a consistency violation under $\mathcal{A}'$ following a symmetric argument.

Our lower bound. We now extend the above proof to randomized protocols. In a randomized protocol, there are two sources of randomness that need to be considered carefully. On one hand, honest nodes can use randomization to their advantage. On the other hand, an adaptive adversary can also leverage randomness. Indeed our lower bound uses a randomized adversarial strategy. In addition, our lower bound crucially relies on the adversary being *strongly adaptive* – the adversary can observe that a message is sent by an honest node h to any other node in a given round r, decide to adaptively corrupt h, and then remove the messages sent by h in round r. We prove the following theorem — here we say that a protocol solves Byzantine Broadcast with probability q iff for any non-uniform p.p.t. strongly adaptive adversary, with probability q, every honest node outputs a bit at the end of the protocol, and consistency and validity are satisfied.

Theorem 4 *If a protocol solves Byzantine Broadcast with $\frac{3}{4} + \epsilon$ probability against a strongly adaptive adversary that can corrupt f nodes ($f < n$), then in expectation, honest nodes collectively need to send at least $(2\epsilon\lfloor f/2 \rfloor)^2$ messages.*

Proof For the sake of contradiction, suppose that a protocol solves Byzantine Broadcast against a strongly adaptive adversary with $\frac{3}{4} + \epsilon$ probability using less than $(2\epsilon\lfloor f/2 \rfloor)^2$ expected messages. This means, regardless of what the adversary does, the protocol errs (i.e., violate either consistency, validity or termination) with no more than $\frac{1}{4} - \epsilon$ probability. We will construct an adversary that makes the protocol err with a probability higher than $\frac{1}{4} - \epsilon$.

Without loss of generality, assume that there exist $\lceil n/2 \rceil$ nodes that output 0 with at most $1/2$ probability if they receive no messages. (Otherwise, there must exist $\lceil n/2 \rceil$ nodes that output 1 with at most $1/2$ probability if they receive no messages, and the entire proof follows from a symmetric argument.) Let V be a set of $\lfloor f/2 \rfloor$ such nodes not containing the designated sender. Note that these nodes may output 1 or they may simply not terminate if they receive no messages. (We can always find such a V because $\lfloor f/2 \rfloor < \lceil n/2 \rceil$). Let U denote the remaining nodes. Let the designated sender send 0.

Next, consider the following adversary $\mathcal{A}$ that corrupts V and makes nodes in V behave honestly except that:

1. Nodes in V do not send messages to each other.
2. Each node in V *ignores* (i.e., pretends that it does not receive) the first $\lfloor f/2 \rfloor$ messages sent to it by nodes in U.

For a protocol to have an expected communication complexity of less than $(2\epsilon\lfloor f/2\rfloor)^2$, honest nodes collectively need to send fewer than that many messages in expectation *regardless of* the adversary's strategy. Let Z be a random variable denoting the number of messages sent by honest nodes to V. We have $E[Z] < (2\epsilon\lfloor f/2\rfloor)^2$. Let X_1 be the event that $Z \leq 2\epsilon\lfloor f/2\rfloor^2$. Then,

$$\Pr[X_1] = \Pr[Z \leq 2\epsilon\lfloor f/2\rfloor^2] \geq \Pr[Z \leq \frac{1}{2\epsilon}E[Z]] > 1 - 2\epsilon.$$

The last step is by Markov's inequality $\Pr[Z > \frac{1}{2\epsilon}E[Z]] < 2\epsilon$.

Let X_2 be the event that among the first $2\epsilon\lfloor f/2\rfloor^2$ messages, a node p picked uniformly at random from V by the adversary receives at most $\lfloor f/2\rfloor$ messages. Observe that among the first $2\epsilon\lfloor f/2\rfloor^2 = 2\epsilon|V|\lfloor f/2\rfloor$ messages, there exist at most $2\epsilon|V|$ nodes that receive more than $\lfloor f/2\rfloor$ of those. Since p is picked uniformly at random from V, $\Pr[X_2] \geq 1 - 2\epsilon$. Thus, we have that

$$\begin{aligned}\Pr[X_1 \cap X_2] &= \Pr[X_1] + \Pr[X_2] - \Pr[X_1 \cup X_2] \\ &> (1 - 2\epsilon) + (1 - 2\epsilon) - 1 = 1 - 4\epsilon.\end{aligned}$$

Now, define another adversary $\mathcal{A}'$ almost identical to $\mathcal{A}$ except that:

1. $\mathcal{A}'$ picks a node $p \in V$ uniformly at random and corrupts everyone else in V except p.
2. $\mathcal{A}'$ blocks the first $\lfloor f/2\rfloor$ attempts that nodes in U send p messages. In other words, whenever some node $s \in U$ attempts to send a message to p in a round, if this is within the first $\lfloor f/2\rfloor$ attempts that nodes in U send p messages, $\mathcal{A}'$ immediately corrupts s (unless s is already corrupted) and removes the message s sends p in that round. Corrupted nodes in U behave honestly otherwise. (In particular, after the first $\lfloor f/2\rfloor$ messages from U to p have been blocked, corrupt nodes behave honestly to p as well.)

Observe that $X_1 \cap X_2$ denotes the event that under adversary $\mathcal{A}$, the total number of messages sent by honest nodes to V is less than $2\epsilon\lfloor f/2\rfloor^2$ and among those, the randomly picked node p has received at most $\lfloor f/2\rfloor$ messages. In this case, p receives no message at all under adversary $\mathcal{A}'$. By the definition of V, p outputs 0 with at most 1/2 probability if it receives no messages. Let Y_1 be the event that p does *not* output 0 under $\mathcal{A}'$. Note that Y_1 includes the event that p outputs 1 as well as the event that p does not terminate. We have $\Pr[Y_1] \geq \Pr[Y_1|X_1 \cap X_2] \cdot \Pr[X_1 \cap X_2] > \frac{1}{2}(1 - 4\epsilon)$.

Meanwhile, we argue that, from the perspective of honest nodes in U, the execution under $\mathcal{A}$ and the execution under $\mathcal{A}'$ are identically distributed. This is because the only difference between the two is that, the first $\lfloor f/2\rfloor$ messages from U to p are intentionally ignored by a corrupt p under $\mathcal{A}$, and blocked using after-the-fact removal under $\mathcal{A}'$.

Under $\mathcal{A}$, honest nodes in U need to output 0 to preserve validity. Recall that the protocol solves Byzantine broadcast with at least $\frac{3}{4} + \epsilon$ probability, so with at least this probability, all honest nodes in U output 0 under $\mathcal{A}$. Let Y_2

be the event that all honest nodes in U output 0 under $\mathcal{A}'$. Since honest nodes in U receive identically distributed messages under $\mathcal{A}$ and $\mathcal{A}'$, $\Pr[Y_2] \geq \frac{3}{4} + \epsilon$.

If Y_1 and Y_2 both occur, then the protocol errs under $\mathcal{A}'$: either consistency or termination is violated. We have

$$\begin{aligned}\Pr[Y_1 \cap Y_2] &= \Pr[Y_1] + \Pr[Y_2] - \Pr[Y_1 \cup Y_2] \\ &> \frac{1}{2}(1 - 4\epsilon) + (\frac{3}{4} + \epsilon) - 1 = \frac{1}{4} - \epsilon.\end{aligned}$$

This contradicts the hypothesis that the protocol solves Byzantine broadcast with $\frac{3}{4} + \epsilon$ probability.

4 Subquadratic BA under Synchrony: $f < (1/3 - \epsilon)n$

This section presents the main ingredients for achieving subquadratic BA. In this section, we opt for conceptual simplicity over other desired properties. In particular, the protocol in this section tolerates only $\frac{1}{3} - \epsilon$ fraction of adaptive corruptions, and completes in $O(\lambda)$ rounds. In the next section, we will show how to improve the resilience to $\frac{1}{2} - \epsilon$ and round complexity to expected $O(1)$.

4.1 Warmup: A Simple Quadratic BA Tolerating 1/3 Corruptions

We first describe an extremely simple quadratic BA protocol, inspired by the Phase-King paradigm [2], that tolerates less than 1/3 corruptions. The protocol proceeds in λ iterations $r = 1, 2, \ldots \lambda$, and every iteration consists of two rounds. For the time being, assume a random leader election oracle that elects and announces a random leader at the beginning of every iteration. 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 iteration). Each iteration r now proceeds as follows where all messages are signed, and only messages with valid signatures are processed:

1. The leader of iteration r flips a random coin b and multicasts $(\texttt{Propose}, r, b)$. Every node i sets $b_i^* := b_i$ if $F = 1$ or if it has not heard a valid proposal from the current iteration's leader. Else, it sets $b_i^* := b$ where b is the proposal heard from the current iteration's leader (if proposals for both $b = 0$ and $b = 1$ have been observed, choose an arbitrary bit).
2. Every node i multicasts $(\texttt{Vote}, r, b_i^*)$. If at least $\frac{2n}{3}$ votes 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 the last iteration, each node outputs the bit that it last voted for.

In short, in every iteration, 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 guarantees consistency unconditionally and achieves termination with overwhelming probability. We now explain why the protocol

works. Henceforth, we refer to a collection of $\frac{2n}{3}$ votes from distinct nodes for the same iteration and the same b as a *certificate* for b.

- *Consistency within an iteration.* Suppose that in iteration r, honest node i observes a certificate for b from a set of nodes denoted S, and honest node j observes a certificate for b' from a 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 iteration exists.* Next, suppose that in some iteration r the leader is honest. We say that this leader chooses a lucky bit b^* iff in iteration $r-1$, no honest node has seen a certificate for $1-b^*$. This means, in iteration r, every honest node either sticks with b^* or switches to the leader's proposal of b^*. Clearly, an honest leader chooses a lucky b^* with probability at least $1/2$. Since leaders are chosen uniformly randomly, except with $\exp(-\Omega(\lambda))$ probability, an honest-leader iteration with a lucky choice exists.
- *Persistence of honest choice after a good iteration.* Now, as soon as we reach an iteration (denoted r) with an honest leader and its choice of bit b^* is lucky, then all honest nodes will vote for b^* in iteration r. Thus all honest nodes will hear certificates for b^* in iteration r; therefore, they will all stick to b^* in iteration $r+1$. By induction, in all future iterations they will stick to b^*.
- *Validity.* 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 iterations.

4.2 Subquadratic Communication through Vote-Specific Eligibility

The above simple protocol requires in expectation linear number of multicast messages (in each round every node multicasts a message). We now consider how to improve the multicast complexity of the warmup protocol. We will also remove the idealized leader election oracle in the process.

Background on VRFs. We rely on a verifiable random function (VRF) [34]. A trusted setup phase is used to generate a public-key infrastructure (PKI): each node $i \in [n]$ obtains a VRF secret key sk_i, and its corresponding public key pk_i. A VRF evaluation on the message μ denoted $(\rho, \pi) \leftarrow \mathsf{VRF}_{\mathsf{sk}_i}(\mu)$ generates a deterministic pseudorandom value ρ and a proof π such that ρ is computationally indistinguishable from random without the secret key sk_i, and with pk_i everyone can verify from the proof π that ρ is evaluated correctly. We use VRF^1 to denote the first output (i.e., ρ above) of the VRF and χ denotes the size of the VRF output, i.e., $\chi = |\mathsf{VRF}_{\mathsf{sk}_i}(\mu)|$.

Strawman: the Chen-Micali approach. We first describe the paradigm of Chen and Micali [12] but we explain it in the context of our warmup protocol. Imagine that now not everyone is required to vote in a round r. Instead, we use the function $\mathsf{VRF}^1_{\mathsf{sk}_i}(\mathtt{Vote}, r) < D$ to determine whether i is eligible to vote in round r where D is a difficulty parameter appropriately chosen such that, in expectation, λ many nodes would be chosen to vote in each round. When

node i sends a `Vote` message, it attaches the VRF's evaluation outcome as well as the proof such that every node can verify its eligibility using its public key pk_i. Correspondingly, when we tally votes, the original threshold $\frac{2n}{3}$ should be changed to $\frac{2\lambda}{3}$, i.e, two-thirds of the expected committee size.

Evaluating the VRF requires knowing the node's secret key. Thus, only the node itself knows at what rounds it is eligible to vote. This may seem to solve the problem because the adversary cannot predict in advance who will be sending messages in every round. The problem with this is that once an adaptive adversary $\mathcal{A}$ notices that some node i was eligible to vote for b in round r (because i just sent a valid vote for b), $\mathcal{A}$ can corrupt i immediately and make i vote for $1-b$ in the same round!

To tackle this precise issue, Chen and Micali [12] relies on the memory-erasure model (referred to as ephemeral keys in their paper) and a *forward-secure* signing scheme. Informally, in a forward secure signing scheme, in the beginning, a node has a key that can sign any messages from any round; after signing a message for round t, the node updates its key to one that can henceforth sign only messages for round $t+1$ or higher, and the round-t secret key should be immediately erased at this point. This way, even if the attacker instantly corrupts a node, it cannot cast another vote in the same round.

Our key insight: bit-specific eligibility. Our key insight is to make the eligibility bit-specific. To elaborate, the committee eligible to vote for b in round r is chosen *independently* from the committee eligible to vote on $1-b$ in the same round. Concretely, node i is eligible to send a `Vote` message for the bit $b \in \{0,1\}$ in round r iff $\mathsf{VRF}^1_{\mathsf{sk}_i}(\texttt{Vote}, r, b) < D$, where D is the aforementioned difficulty parameter.

What does this achieve? Suppose that the attacker sees some node i votes for the bit b in round r. Although the attacker can now immediately corrupt i, 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 the same round. Thus, corrupting i is no more useful to the adversary than corrupting any other node.

Finally, since we already make use of the VRF, as a by-product we can remove the idealized leader election oracle in the warmup protocol: a node i is eligible for making a proposal in iteration r iff $\mathsf{VRF}^1_{\mathsf{sk}_i}(\texttt{Propose}, r, b) < D_0$ where D_0 is a separate difficulty parameter explained below. Naturally, the node attaches the VRF evaluation outcome and proof with its proposal so that others can verify its eligibility.

Difficulty parameters. The two difficulty parameters D and D_0 need to be specified differently. Recall that D is used to elect a committee in each round for sending `Vote` messages; and D_0 is used for leader election.

1. D should be set such that each committee is λ-sized in expectation. If we think of a VRF output as a uniformly random number between 0 and 1, then we should set $D = \lambda/n$.

2. D_0 should be set such that there is a constant probability of having a unique leader who is honest. To be concrete, if we again think of a VRF output as a uniformly random number between 0 and 1, then we can set $D_0 = 1/n$.

Since we are interested in making communication scale better with n, we assume $n > \lambda$; otherwise, one should simply use the quadratic protocol.

Putting it together. More formally, 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 iteration r and if so, it multicasts $(\mathtt{T}, r, b, i, \rho, \pi)$, where $\mathtt{T} \in \{\mathtt{Propose}, \mathtt{Vote}\}$ stands for the type of the message, ρ is a pseudorandom evaluation result, and π is a proof proving that i indeed is eligible. Now, our new committee-sampling based subquadratic protocol is almost identical to the warmup protocol except for the following changes:

- every occurrence of multicast is now replaced with "conditionally multicast";
- the threshold of certificates (i.e., number of votes for a bit to stick) is now $\frac{2\lambda}{3}$; and
- upon receiving every message, a node checks the proof to verify the sender's eligibility to send that message.

4.3 Proof Sketch

To help our analysis, we shall abstract away the cryptography needed for eligibility election, and instead think of eligibility election as making queries to a trusted party called $\mathcal{F}_{\text{mine}}$. We call an attempt for node i to check its eligibility to send either a `Propose` or `Vote` message a *mining* attempt for a `Propose` or `Vote` message (inspired by Bitcoin's terminology where miners "mine" blocks). Specifically, if a node i wants to check its eligibility for sending $(\mathtt{T}, r, b)$ where $\mathtt{T} \in \{\mathtt{Propose}, \mathtt{Vote}\}$, 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 – this is used in place of verifying the VRF proof. If a so-far-honest node makes a mining attempt for some $(\mathtt{T}, r, b)$, it 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*.

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.

Consistency within an iteration. We first argue why "consistency within an iteration" still holds with the new protocol. There are at most $(\frac{1}{3} - \epsilon)n$ corrupt nodes, each of which might try to mine for two votes (one for each bit) in every iteration r. On the other hand, each so-far-honest node will try to mine for only one vote in each iteration. Therefore, in iteration r, the total number of mining attempts (honest and corrupt) for `Vote` messages is at most

$2(\frac{1}{3}-\epsilon)n + (\frac{2}{3}+\epsilon)n = (\frac{4}{3}-\epsilon)n$, each of which is **independently** successful with probability $\frac{\lambda}{n}$. Hence, if there are $\frac{2\lambda}{3}$ votes 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 $\exp(-\Omega(\lambda))$ probability, by the Chernoff bound. Therefore, except with $\exp(-\Omega(\lambda))$ probability, if any node sees $\frac{2\lambda}{3}$ votes for some bit b, then no other node sees $\frac{2\lambda}{3}$ votes for a different bit b'.

A good iteration exists. We now argue why "a good iteration exists" in our new protocol. Here, for an iteration 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; and 2) if some honest nodes want to stick to a bit b^* in iteration r, the leader's random coin must agree with b^*. (Note that if multiple so-far-honest nodes successfully mine `Propose` messages, this iteration is not a good iteration). Every so-far-honest node makes only one `Propose` mining attempt per iteration. Every already corrupt node can make two `Propose` mining attempts in an iteration, one for each bit. Since our `Propose` mining difficulty parameter D_0 is set such that each attempt succeeds with $1/n$ probability, in every iteration, 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 λ iterations, a good iteration exists except with $\exp(-\Omega(\lambda))$ probability,

Remainder of the proof. Finally, "persistence of honest choice after a good iteration" and "validity" hold in a relatively straightforward fashion by applying the standard Chernoff bound.

Remark. We stress that for the above argument to hold, it is important that the eligibility be tied to the bit being proposed/voted. Had it not been the case, the adversary could observe whenever an honest node sends $(\mathtt{T}, r, b)$, and immediately corrupt the node in the same round and make it send $(\mathtt{T}, r, 1-b)$ too. If `T` is `Vote`, whenever there are $\frac{2\lambda}{3}$ votes for b in iteration r, by corrupting all these nodes that are eligible to vote, the adversary can construct $\frac{2\lambda}{3}$ votes for $1-b$, and thus "consistency within an iteration" does not hold. If `T` is `Propose`, whenever there is a so-far-honest leader in iteration r, the adversary can corrupt this leader and make it also propose the opposite value; this way, every leader would equivocate and no good iteration would exist.

5 Subquadratic BA under Synchrony: $f < (1/2 - \epsilon)n$

In this section, we present our synchronous BA protocol that achieves expected subquadratic communication complexity (expected sublinear multicast complexity) and expected constant round complexity, and tolerates $f < (\frac{1}{2}-\epsilon)n$ adaptive corruptions where ϵ is a constant. Our starting point is Abraham et al. [1], a synchronous quadratic BA protocol tolerating $f < n/2$ corruptions. We explain Abraham et al. at a high level and then apply the techniques

introduced in the previous section to achieve subquadratic communication complexity.

5.1 Warmup: Quadratic BA Tolerating 1/2 Corruptions

Our description below assumes $n = 2f + 1$ nodes in total. The protocol runs in iterations $r = 1, 2, \ldots$ Each iteration has four synchronous rounds called `Status`, `Propose`, `Vote`, and `Commit`, respectively. Messages sent at the beginning of a round will be received before next round. All messages are signed. Henceforth, a collection of $f+1$ (signed) iteration-r `Vote` messages for the same bit $b \in \{0, 1\}$ from distinct nodes is said to be an *iteration-r certificate* for b. A certificate from a higher iteration is said to be a higher certificate. For the time being, assume a random leader election oracle that elects a random leader L_r at the beginning of every iteration r.

Below is the protocol for an iteration $r \geq 2$. The protocol for the very first iteration $r = 1$ skips the `Status` and `Propose` rounds.

1. `Status`. Every node multicasts a `Status` message of the form $(\texttt{Status}, r, b, \mathcal{C})$ containing the highest certified bit b it has seen so far as well as the corresponding certificate $\mathcal{C}$.
2. `Propose`. The leader L_r chooses a bit b with a highest certificate denoted $\mathcal{C}$ breaking ties arbitrarily. The leader multicasts $(\texttt{Propose}, r, b, \mathcal{C})$. To unify the presentation, we say that a bit b without any certificate has an iteration-0 certificate and it is treated as the lowest ranked certificate.
3. `Vote`. For the very first iteration $r = 1$, a node votes for its input bit b by multicasting $(\texttt{Vote}, r = 1, b)$.
   For all iterations $r \geq 2$, if a validly signed $(\texttt{Propose}, r, b, \mathcal{C})$ message has been received from L_r with a certificate $\mathcal{C}$ for b, and if the node has not observed a *strictly* higher certificate for $1 - b$, it multicasts an iteration-r `Vote` message for b of the form $(\texttt{Vote}, r, b)$ with the leader's proposal attached. Importantly, if the node has observed a certificate for the opposite bit $1 - b$ from the same iteration as $\mathcal{C}$, it *will* vote for b.
4. `Commit`. If a node has received $f + 1$ iteration-r signed votes for the same bit b from distinct nodes (which form an iteration-r certificate $\mathcal{C}$ for b) and no iteration-r vote for $1 - b$, it multicasts an iteration-r `Commit` message for b of the form $(\texttt{Commit}, r, b)$ with the certificate $\mathcal{C}$ attached.

⋆ (This step is not part of the iteration and can be executed at any time.) If a node has received $f + 1$ `Commit` messages for the same b from the same iteration from distinct nodes, it multicasts a termination message of the form $(\texttt{Terminate}, b)$ with the $f + 1$ `Commit` messages attached. The node then outputs b and terminates. This last message will make all other honest nodes multicast the same `Terminate` message, output b and terminate in the next round.

Consistency. The protocol achieves consistency due to the following key property. *If an honest node outputs a bit b in iteration r, then no certificate*

for $1-b$ can be formed in iteration r and all subsequent iterations. We explain why this property holds below.

An honest node outputs b in iteration r, only if it has observed $f+1$ iteration-r `Commit` messages (from distinct nodes) for b. One of these must have been sent by an honest node henceforth indexed by i^*. For an iteration-r certificate for $1-b$ to exist, an honest node must have multicast a vote for $1-b$. But in that case, i^* would have received this conflicting vote and thus would not have sent the commit message for b. We have reached a contradiction. Thus, we can rule out any iteration-r certificate for $1-b$.

Furthermore, by the end of iteration r, all nodes will receive from node i^* an iteration-r certificate for b. Since no iteration-r certificate for $1-b$ exists, no honest node votes for $1-b$ in iteration $r+1$; hence, no iteration-$(r+1)$ certificate for $1-b$ can come into existence; hence no honest node votes for $1-b$ in iteration $r+2$, and so on. The preference for a higher certificate ensures consistency for all subsequent iterations following a simple induction.

Validity. Recall that the first iteration skips `Status` and `Propose` and directly starts with `Vote`. If all honest nodes have the same input bit b, then they all vote for b in the first iteration. By the end of the first iteration, every honest node has an iteration-1 certificate for b and no iteration-1 certificate for $1-b$ exists. Validity then follows from consistency.

Expected constant round complexity. Once an iteration has an honest leader, it will sign a unique proposal for the bit b with the highest certificate reported by honest nodes. Then, all honest nodes send `Vote` and `Commit` messages for b, output and terminate in that iteration. Since leaders are selected at random, in expectation, an honest leader emerges in two iterations.

5.2 Subquadratic Communication through Vote-Specific Eligibility

The above simple protocol requires quadratic communication (in each round every node multicasts a message). We now improve the communication complexity to subquadratic and we will also remove the idealized leader election oracle in the process.

We now use the vote-specific eligibility to determine for each iteration, who is eligible for sending `Status`, `Propose`, `Vote` and `Commit` messages for 0 and 1, respectively. To keep the presentation simple, we abstract away the cryptographic primitives for eligibility election and model it as an ideal functionality $\mathcal{F}_{\text{mine}}$. As before, we call an attempt for node i to check eligibility to send a message a *mining* attempt. Concretely, node i is eligible to send $(\mathtt{T}, r, b)$ where $\mathtt{T}$ is `Status`, `Vote`, or `Commit`, iff

$$\mathcal{F}_{\text{mine}}.\mathtt{mine}(i, \mathtt{T}, r, b) < D,$$

node i is eligible to send $(\mathtt{Terminate}, b)$ iff

$$\mathcal{F}_{\text{mine}}.\mathtt{mine}(i, \mathtt{Terminate}, b) < D,$$

and node i is eligible to send $(\texttt{Propose}, r, b)$ iff

$$\mathcal{F}_{\text{mine}}.\texttt{mine}(i, \texttt{Propose}, r, b) < D_0.$$

D and D_0 are appropriate *difficulty* parameters such that each mining attempt for `Status`/`Vote`/`Commit`/`Terminate` has a λ/n probability to be successful and each mining attempt for leader proposal has a $1/n$ probability to be successful. As before, we assume $n > \lambda$; otherwise, one should simply use the quadratic protocol.

We use the phrase "node i *conditionally multicasts* a message" to mean that node i checks with $\mathcal{F}_{\text{mine}}$ if it is eligible to send that message and only multicasts the message if it is. Now, the committee-sampling based subquadratic protocol is almost identical to the warmup protocol except for the following changes:

- every occurrence of multicast is now replaced with "conditionally multicast";
- every occurrence of $f + 1$ `Vote` or `Commit` messages is now replaced with $\lambda/2$ messages of that type [1] ; and
- upon receiving a message of the form (i, m) (including messages attached with other messages), a node invokes $\mathcal{F}_{\text{mine}}.\texttt{verify}(i, \mathsf{m})$ to verify node i's eligibility to send that message. Note that m can be of the form $(\mathtt{T}, r, b)$ where $\mathtt{T} \in \{\texttt{Status}, \texttt{Propose}, \texttt{Vote}, \texttt{Commit}\}$ or of the form $(\texttt{Terminate}, b)$.

5.3 Proof

We prove our new protocol works in this subsection. The proofs mostly follow the sketch in Section 5.1 — except that we now need to analyze the stochastic process induced by eligibility. Our stochastic analysis here is performed assuming an idealized $\mathcal{F}_{\text{mine}}$, and this idealized oracle will be removed later in Section 10.

We will frequently use the standard Chernoff bound, given below.

Lemma 1 (Chernoff) *Let $X = \sum_{i=1}^{n} X_i$ be the sum of n independent Boolean random variables and μ be the expectation of X. For $0 < \delta < 1$,*

$$\Pr[X \leq (1-\delta)\lambda] < e^{-\frac{\delta^2 \mu}{2}},$$
$$\Pr[X \geq (1+\delta)\lambda] < e^{-\frac{\delta^2 \mu}{3}}.$$

To prove consistency and validity, we first establish the following lemma.

Lemma 2 *Except for $\exp(-\Omega(\lambda))$ probability, for any `Status`/`Vote`/`Commit` message for bit b in iteration r, (i) less than $\lambda/2$ eventually-corrupt nodes are eligible to send it, and (ii) more than $\lambda/2$ forever-honest nodes are eligible to send it.*

[1] If a node receives $\lambda/2$ `Commit` messages but is not eligible to send a `Terminate` message, it terminates without sending one.

Proof By our choice of D, each node independently has a λ/n probability to send the said message. For part (i), recall that the adversary can make at most $(1/2-\epsilon)n$ adaptive corruptions where $0 < \epsilon < 1/2$ is a constant. Thus, in expectation, $(1/2-\epsilon)\lambda$ eventually-corrupt nodes are eligible to send the said message. Plugging $\mu = (1/2-\epsilon)\lambda$ and $\delta = \frac{\lambda/2}{(1/2-\epsilon)\lambda} - 1 = \frac{\epsilon}{1/2-\epsilon} > 0$ into the Chernoff upper tail bound in Lemma 1, the probability that no less than $\lambda/2$ eventually-corrupt nodes are eligible to send the said message is bounded by $\exp(-\Theta(\epsilon^2\lambda))$.

Part (ii) is similar. In expectation, $(1/2+\epsilon)\lambda$ forever-honest nodes are eligible to send the said message. Plugging $\mu = (1/2+\epsilon)\lambda$ and $\delta = 1 - \frac{\lambda/2}{(1/2+\epsilon)\lambda} = \frac{\epsilon}{1/2+\epsilon}$ into the Chernoff lower tail bound in Lemma 1, the probability that no more than $\lambda/2$ forever-honest nodes are eligible to send the said message is bounded by $\exp(-\Theta(\epsilon^2\lambda))$.

Theorem 5 (Consistency) *Except for* $\exp(-\Omega(\lambda))$ *probability, if an honest node outputs a bit* b *in iteration* r*, then no certificate for* $1-b$ *can be formed in iteration* r *and all subsequent iterations.*

Proof An honest node outputs b in iteration r, only if it has observed $\lambda/2$ `Commit` messages for b. By Lemma 2, except for $\exp(-\Omega(\lambda))$ probability, not all of them are sent by eventually-corrupt nodes; in other words, one of the `Commit` messages was sent by a forever-honest node henceforth indexed by i^*. Similarly, for an iteration-r certificate for $1-b$ to exist, except for $\exp(-\Omega(\lambda))$ probability, one forever-honest node has multicast a vote for $1-b$. But in that case, i^* would have received this conflicting vote and would not have sent the `Commit` message for b. We have reached a contradiction. Thus, no iteration-r certificate for $1-b$ exists except for $\exp(-\Omega(\lambda))$ probability.

Furthermore, by the beginning of iteration $r+1$, all forever-honest nodes will receive from node i^* an iteration-r certificate for b. The lack of iteration-r certificate for $1-b$ together with the preference to higher certificate ensures that no forever-honest node will vote for $1-b$ in iteration $r+1$. To form a certificate for $1-b$ in a subsequent iteration, all $\lambda/2$ votes have to come from eventually-corrupt nodes, which happens with $\exp(-\Omega(\lambda))$ probability by Lemma 2. An induction then completes the proof.

Theorem 6 (Validity) *Except for* $\exp(-\Omega(\lambda))$ *probability, if all honest nodes have the same input bit* b*, then all nodes will output* b*.*

Proof Straightforward from Lemma 2: in the first iteration, except for the said probability, there will be sufficient forever-honest nodes to send $(\texttt{Vote}, r=1, b)$, and there will not be sufficient eventually-corrupt nodes to vote for $1-b$. Validity then follows from consistency.

We then turn to liveness and expected round/communication/multicast complexity. We say an iteration r is a good iteration if a *single* so-far-honest node successfully mines a `Propose` message, and no already-corrupt node successfully mines a `Propose` message. (Note that if multiple so-far-honest nodes successfully mine `Propose` messages, this iteration is not a good iteration).

Lemma 3 *If fewer than $\epsilon n/2$ forever-honest nodes have terminated, then, every iteration independently has a $\Theta(1)$ probability to be a good iteration.*

Proof In any fixed iteration r, suppose there are n_h so-far-honest nodes that have not terminated and n_c already-corrupt nodes. Each so-far-honest node makes one attempt to propose (either 0 or 1), whereas each already-corrupt node can make two attempts to propose (both 0 and 1). Recall that we set D_0 such that each mining attempt for `Propose` succeeds with probability $1/n$.

The probability that exactly one honest `Propose` attempt succeeds and no corrupt `Propose` attempt succeeds is $\binom{n_h}{1}\frac{1}{n}(1-\frac{1}{n})^{n_h-1+2n_c}$. Observe that $n_h + n_c < n$, $n_c < n/2$ and $n_h > n/2$, so the above expression is greater than $\frac{1}{2} \cdot (1-\frac{1}{n})^{1.5n} = \Theta(1)$.

Lemma 4 *Except for $\exp(-\Omega(\lambda))$ probability, if at least $\epsilon n/2$ forever-honest nodes have terminated, all so-far-honest nodes terminate by the end of the next round.*

Proof Each of the $\epsilon n/2$ forever-honest nodes attempts to send `Terminate`, and each has a λ/n probability to be eligible. The probability that none of them is eligible is $(1-\lambda/n)^{\epsilon n/2} < \exp(-\epsilon\lambda/2) = \exp(-\Omega(\lambda))$. Note that the adversary can fully control in what order honest nodes terminate, but it cannot predict which honest nodes are eligible to send `Terminate`. Thus, it cannot bias the above probability. Except for this exponentially small probability, at least one so-far-honest node who have terminated is eligible to send a `Terminate` message, which makes all so-far-honest nodes terminate by the end of the next round.

Lemma 5 *When there is a good iteration, all so-far-honest nodes terminate at the end of that iteration except for $\exp(\Omega(-\lambda))$ probability.*

Proof For any iteration, if at least $\epsilon n/2$ forever-honest nodes have terminated, then by Lemma 4, all so-far-honest nodes terminate by the end of the next round except for $\exp(\Omega(-\lambda))$ probability.

Else, by Lemma 3, with at least $\Theta(1) - \exp(-\Omega(\lambda)) = \Theta(1)$ probability, a single so-far-honest node sends `Propose` and no already-corrupt node sends `Propose`. Further, since at least $(\frac{1}{2} + \frac{\epsilon}{2})n$ so-far-honest nodes have not terminated, by Lemma 2, $\lambda/2$ so-far-honest nodes are eligible to send `Status`/`Vote`/`Commit` messages except for $\exp(\Omega(-\lambda))$ probability. Thus, all so-far-honest nodes terminate except for $\exp(\Omega(-\lambda))$ probability.

Theorem 7 (Liveness) *Except for $\exp(-\Omega(\lambda))$ probability, all honest nodes terminate in $O(\lambda)$ rounds.*

Proof The probability that none of the λ iterations is good is $(1-\Theta(1))^{\lambda} = \exp(-\Omega(\lambda))$. The theorem then follows from Lemma 3, 4 and 5.

In the following statements, we use χ to denote the message size incurred for each message sent by a node once we instantiate $\mathcal{F}_{\text{mine}}$.

Theorem 8 (Efficiency) *In expectation, all honest nodes terminate in $O(1)$ rounds and collectively send $O(n\lambda\chi)$ bits (i.e., $O(\lambda\chi)$ multicast complexity).*

Proof The expected constant round complexity follows from Lemma 3, 4 and 5. In each round, in expectation, at most λ so-far-honest nodes multicast messages where each message is of size χ bits. Thus, honest nodes collectively send $O(n\lambda\chi)$ bits in expectation.

Corollary 1 (Efficiency) *Except for $\exp(-\Omega(\lambda))$ probability, all honest nodes terminate in $O(\lambda)$ rounds and collectively send $O(n\lambda^2\chi)$ bits (i.e., $O(\lambda^2\chi)$ multicast complexity).*

Proof The probability that none of the λ iterations is good is $(1-\Theta(1))^\lambda = \exp(-\Omega(\lambda))$. By Lemma 2 and a union bound, except for $\exp(-\Omega(\lambda))$ probability, in each round of each iteration, $O(\lambda)$ so-far-honest nodes send messages of size χ bits, leading to $O(n\lambda\chi)$ bits in total.

Theorem 9 *For any constant $0 < \epsilon < 1/2$, the protocol in this section solves Byzantine agreement with $1-\exp(-\Omega(\lambda))$ probability; the protocol terminates in expected $O(1)$ rounds, and honest nodes collectively send $O(\lambda\chi)$ bits in expectation.*

Proof Follows from Theorem 5, 6, 7 and 8.

6 Subquadratic BA under Partial Synchrony

In this section, we describe a partially synchronous BA protocol that tolerates $\frac{1}{3} - \epsilon$ adaptive corruptions.

6.1 Warmup: Communication-Inefficient Underlying BA

Our starting point is a simple quadratic partially synchronous BA protocol in the unknown Δ model [19]. The protocol is, in fact, similar to the 4-round-per-iteration synchronous warmup protocol in Section 5.1. The protocol also runs in iterations and each iteration consists of four steps. The key change is that every λ iterations, all nodes double the step length. At some point the step length will reach or exceed Δ rounds. Until then, messages may or may not arrive within the step. But after that point, messages sent at the beginning of a step will be received before the next step, and the protocol will terminate within λ iterations with high probability.

Our description below assumes $n = 3f + 1$ nodes in total. The protocol runs in iterations $r = 1, 2, \ldots$. Each iteration runs in four steps, called `Status`, `Propose`, `Vote`, and `Commit`, respectively. At the beginning, each step will be of length 1 round. Every λ iterations, all nodes double the length of a step. All messages are signed. We denote a collection of $2f + 1$ (signed) iteration-r `Vote` messages for the same bit $b \in \{0, 1\}$ from distinct nodes as an *iteration-r*

certificate for b (in comparison, the synchronous protocol in Section 5.1 required only $f+1$ votes). A certificate from a higher iteration is said to be a higher certificate. For the time being, assume a random leader election oracle that elects a random leader L_r at the beginning of every iteration r.

1. `Status`. Every node multicasts a `Status` message of the form $(\texttt{Status}, r, b, \mathcal{C})$ containing the highest certified bit b it has seen so far as well as the corresponding certificate $\mathcal{C}$. In the first iteration $r = 1$, it will send its signed input bit to the leader.
2. `Propose`. The leader L_r chooses a bit b with a highest certificate denoted $\mathcal{C}$. The leader multicasts $(\texttt{Propose}, r, b, \mathcal{C})$. If the leader does not have a certificate of size $2f+1$, it sends a certificate $\mathcal{C}$ of size $\geq f+1$ each of which is a signed input bit. We call the latter certificate of size $\geq f+1$ an input certificate and it is the lowest ranked certificate.
3. `Vote`. If a validly signed $(\texttt{Propose}, r, b, \mathcal{C})$ message has been received from L_r with a certificate $\mathcal{C}$ for b, and if the node has not observed a higher certificate for $1-b$, it multicasts an iteration-r `Vote` message for b of the form $(\texttt{Vote}, r, b)$ with the leader's proposal attached.
4. `Commit`. If a node has received $2f+1$ iteration-r signed votes for the same bit b from distinct nodes (which form an iteration-r certificate $\mathcal{C}$ for b), it multicasts an iteration-r `Commit` message for b of the form $(\texttt{Commit}, r, b)$ with the certificate $\mathcal{C}$ attached.

$\star$ (This step is not part of the iteration and can be executed at any time.) If a node has received $2f+1$ `Commit` messages for the same b from the same iteration from distinct nodes, it multicasts a termination message of the form $(\texttt{Terminate}, b)$ with the $2f+1$ `Commit` messages attached. The node then outputs b and terminates. This last message will make all other honest nodes multicast the same `Terminate` message, output b and terminate.

Consistency. The analysis for consistency follows similar arguments to Section 5.1. We show that *if any honest node outputs a bit b in iteration r, then no certificate for $1-b$ can be formed in iteration r and all subsequent iterations,* assuming ideal signatures.

If an so-far-honest node outputs b in iteration $r \geq 1$, there must be an iteration-r certificate for b. Recall that a certificate in this protocol consists of $2f+1$ `Vote` messages from that iteration. For an iteration-r certificate for $1-b$ to also exist, $f+1$ nodes need to vote for both b and $1-b$. But there are only f corrupted nodes, a contradiction. Thus, we can rule out any iteration-r certificate for $1-b$.

Furthermore, $2f+1$ nodes have sent `Commit` messages for b in iteration r. This means at least $f+1$ honest nodes have seen the iteration-r certificate for b. The preference for a higher certificate then ensures consistency for all subsequent iterations. Since no iteration-r certificate for $1-b$ exists, those $f+1$ honest nodes will not vote for $1-b$ in iteration $r+1$; hence, no iteration-$(r+1)$ certificate for $1-b$ can come into existence; hence, those $f+1$ honest nodes

will not vote for $1-b$ in iteration $r+2$ and so on. A simple induction completes the proof.

Validity. Observe that the protocol starts with signed inputs sent to the leader. Before any higher ranked certificate can be formed, the leader needs to send an input certificate of $f+1$ signed inputs for the same value in its proposal. If all honest nodes start with the same value b, no leader can create $\geq f+1$ sized input certificate for value $1-b$, proving validity.

Termination. Once there is an honest leader after length of a step $\geq \Delta$ rounds, all honest nodes send to and receive from each other `Vote` and `Commit` messages, output and terminate in that iteration. Since leaders are selected at random, in expectation, an honest leader emerges in $O(1)$ iterations once the step size exceeds Δ. Since the step lengths double every λ iterations, the protocol will terminate in expected $O(\lambda\Delta)$ rounds.

6.2 Partially Synchronous Subquadratic BA

Bit-specific eligibility can be added in a fashion similar to Section 5.2. Node i is eligible to send `Status`, `Vote`, `Commit`, or a `Terminate` message with difficulty D, and is eligible to send `Propose` with difficulty D_0. All eligibility depends on the bit $b \in \{0,1\}$ and (with the exception of `Terminate`) the iteration number r. D and D_0 are appropriate *difficulty* parameters such that each of the multicast messages except a proposal has a λ/n probability to be eligible and each leader proposal has a $1/n$ probability to be eligible. As before, we assume $n > \lambda$; otherwise, one should simply use the quadratic protocol.

Now, the subquadratic protocol is almost identical to the warmup protocol except for the following changes:

- every occurrence of multicast is now replaced with "conditionally multicast";
- every occurrence of $2f+1$ `Vote` or `Commit` messages is now replaced with $2\lambda/3$ messages of that type;
- every occurrence of an input certificate is now replaced with $\geq \lambda/3$ messages of that type;
- upon receiving a message of the form (i, m) (including messages attached with other messages), a node invokes $\mathcal{F}_{\text{mine}}.\texttt{verify}(i, \mathsf{m})$ to verify node i's eligibility to send that message. Note that m can be of the form $(\mathtt{T}, r, b)$ where $\mathtt{T} \in \{\texttt{Status}, \texttt{Propose}, \texttt{Vote}, \texttt{Commit}\}$ or of the form $(\texttt{Terminate}, b)$.

6.3 Proof

The proofs mostly follow the sketch in Section 6.1 and the stochastic process in Section 5.3. As before, our analysis here assumes an idealized $\mathcal{F}_{\text{mine}}$, and account for its failure probabilities later in Section 10.

Theorem 10 (Consistency) *If an honest node outputs a bit b in iteration r, then no certificate for $1-b$ can be formed in iteration r and all subsequent iterations, except for* $\exp(-\Omega(\lambda))$ *probability.*

Proof An honest node outputs b in iteration r, only if it has observed $2\lambda/3$ `Vote` messages and $2\lambda/3$ `Commit` messages for b.

If an iteration-r certificate for $1-b$ also exists, then there are at least $4\lambda/3$ successful mining attemps on `Vote` in iteration r. There are at most $(\frac{1}{3}-\epsilon)n$ eventually-corrupt nodes; each of them may attempt to vote for both bits. All remaining nodes are forever-honest and will only attempt to vote for one bit. Thus, there are at most $(\frac{4}{3}-\epsilon)n$ total mining attempts, each with a probability of λ/n to be successful but at least $4\lambda/3$ attempts succeeded. This happens with $\exp(-\Omega(\lambda))$ probability: simply plug $\mu=(\frac{4}{3}-\epsilon)\lambda$ and $\delta=\frac{4\lambda/3}{(4/3-\epsilon)\lambda}-1$ into the Chernoff upper tail bound in Lemma 1.

Furthermore, if some forever-honest node receives $2\lambda/3$ `Commit` messages, it implies that $(\frac{2}{3}-\epsilon')n$ nodes have sent `Commit` messages except for $\exp(-\Omega(\lambda))$ probability, due to the Chernoff lower tail bound in Lemma 1. At least $(\frac{1}{3}-\epsilon'+\epsilon)n$ of these nodes are forever-honest. In iteration $r+1$, the remaining $(\frac{2}{3}+\epsilon)-(\frac{1}{3}-\epsilon'+\epsilon)=(\frac{1}{3}+\epsilon')$ may mine for $1-b$ (since if a forever-honest node has sent a `Commit` message for b in iteration r, it will not attempt to vote $1-b$ in iteration $r+1$). In addition, $(\frac{1}{3}-\epsilon)n$ fraction of corrupt nodes mine for $1-b$. To form a certificate for $1-b$ in iteration $r+1$, $2\lambda/3$ out of the $(\frac{2}{3}+\epsilon'-\epsilon)n$ mining attempts for $(\texttt{Vote}, r+1, 1-b)$ from eventually-corrupt nodes need to succeed. Picking $\epsilon'=\epsilon/2$, by the Chernoff upper tail bound, this happens with $\exp(-\Omega(\lambda))$ probability. An induction then completes the proof.

Theorem 11 (Validity) *If all honest nodes have the same input bit b, then all nodes will eventually output b, except for* $\exp(-\Omega(\lambda))$ *probability.*

Proof By a Chernoff bound, except for the said probability, there will be fewer than $\lambda/3$ signed input bits for $1-b$ (from eventually-corrupt nodes) in any iteration. So there is no certificate (input certificate or normal certificate) for $1-b$ and hence, $1-b$ will never be proposed. Validity then follows from safety.

We now prove liveness and efficiency. Lemma 3 from Section 5.3 still applies. Lemma 4 and 5 need very minor modifications as follows but their proofs remain almost identical (so we skip them).

Lemma 6 *Except for* $\exp(-\Omega(\lambda))$ *probability, if at least $\epsilon n/2$ forever-honest nodes have terminated, all so-far-honest nodes terminate within Δ rounds.*

Lemma 7 *When the length of a step is at least Δ rounds and there is a good iteration, all so-far-honest nodes terminate at the end of that iteration except for* $\exp(\Omega(-\lambda))$ *probability.*

Theorem 12 (Liveness) *Except for* $\exp(-\Omega(\lambda))$ *probability, all honest nodes terminate in $O(\lambda\Delta)$ rounds.*

Proof It takes at most at most $\log \Delta$ times for the step length to increase to at least Δ rounds. Hence, by then, there have been at most $\lambda \log \Delta$ iterations, and at most $4\lambda(\Delta + \Delta/2 + \cdots + 2 + 1) + 1 = O(\lambda\Delta)$ rounds have passed.

By Lemma 7, once the length of a step is at least Δ rounds, the protocol terminates in the first good iteration. By Lemma 3, each iteration independently has a constant probability to be a good iteration. The probability that none of the λ iterations is good is $(1 - \Theta(1))^\lambda = \exp(-\Omega(\lambda))$. Thus, all so-far-honest nodes terminate in $O(\lambda\Delta)$ rounds except for $\exp(-\Omega(\lambda))$ probability.

The efficiency analysis are similar to Section 5.

In the following statements, we use χ to denote the message size incurred for each message sent by a node once we instantiate $\mathcal{F}_{\text{mine}}$.

Theorem 13 (Efficiency) *All honest nodes terminate in expected $O(\lambda\Delta)$ rounds and collectively send $O(n\chi\lambda^2 \log \Delta)$ bits in expectation (i.e., $O(\chi\lambda^2 \log \Delta)$ multicast complexity).*

Proof As proved in Theorem 12, by the time when the step length has been increased to at least Δ rounds, there has been at most $O(\lambda\Delta)$ rounds. After that, the protocol terminates in expected constant iterations, which are $O(\Delta)$ rounds. Thus, all honest nodes terminate in expected $O(\lambda\Delta)$ rounds. Each iteration has four steps. In each step of each iteration, in expectation, so-far-honest nodes send $O(\lambda)$ messages of size χ. Thus, honest nodes collectively send $O(n\chi\lambda^2 \log \Delta)$ bits in expectation (i.e., $O(\chi\lambda^2 \log \Delta)$ multicast complexity).

Corollary 2 (Efficiency) *Except for $\exp(-\Omega(\lambda))$ probability, all honest nodes terminate in $O(\lambda\Delta)$ rounds and collectively send $O(n\chi\lambda^2 \log \Delta)$ bits (i.e., $O(\chi\lambda^2 \log \Delta)$ multicast complexity).*

Proof Similar to that of Corollary 1.

Theorem 14 *For any constant $0 < \epsilon < 1/3$, the protocol in this section solves Byzantine agreement with $1 - \exp(-\Omega(\lambda))$ probability; the protocol terminates in expected $O(\lambda\Delta)$ rounds, and honest nodes collectively send $O(\chi\lambda^2 \cdot \log \Delta)$ bits in expectation.*

Proof Follows from Theorems 10, 11, 12 and 13.

Remark. Since we employ cryptography and assume computationally-bounded adversaries, our network model for partial synchrony is adopted from CKPS [7]. Specifically, we assume that (i) honest nodes send polynomially many messages (or Δ is polynomially bounded), and (ii) the delivery of messages is controlled by an adversary.

7 Necessity of Setup Assumptions for Sublinear Multicast Complexity

In this section, we show that some form of setup assumption is needed for multicast-based subquadratic BA. Specifically, with plain authenticated channels, we show the impossibility of sublinear multicast-complexity BA. In this

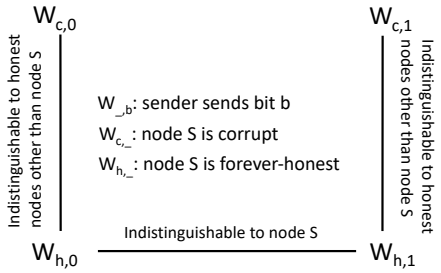

Fig. 1 Relationships between different worlds in the sublinear multicast complexity without setup assumptions.

model, a message carries the true identity of the sender, i.e., the communication channel authenticates the sender, but no other setup is available.

As mentioned in Section 2, proving the lower bound for Byzantine broadcast makes it stronger (and applicable to BA). Thus, we restate the lower bound (i.e., Theorem 3) for Byzantine broadcast below.

Theorem 15 *In a plain authenticated channel model without setup assumptions, no protocol can solve Byzantine broadcast with C multicast complexity with probability $p > 5/6$ under C adaptive corruptions.*

Our proof is inspired by the classical techniques for proving consensus lower bounds in the authenticated channel model [21, 32, 33]; however, we extend known techniques in novel manners, particularly in the way we rely on the ability to make adaptive corruptions to complete the proof. Our proof assumes that the protocol works in synchronous rounds.

Proof Suppose for the sake of contradiction that there exists a protocol that solves Byzantine broadcast using C multicast complexity with probability $p > 5/6$, in the authenticated channel model without any trusted setup, and tolerating C adaptive corruptions.

We focus on a special node S that is not the designated sender. We consider four worlds: $W_{c,0}$, $W_{c,1}$, $W_{h,0}$, and $W_{h,1}$. In world $W_{c,*}$, node S is corrupt whereas in world $W_{h,*}$, node S is forever-honest. The designated sender sends bit b in world $W_{*,b}$.

The high-level structure of the proof is depicted in Figure 1. First, the designated sender is honest in $W_{c,b}$. Thus, with probability $p > 5/6$, honest nodes output b in $W_{c,b}$ to preserve validity. Next, we will show that world $W_{c,b}$ and world $W_{h,b}$ are indistinguishable to nodes that are forever-honest in both. Hence, with probability $p > 5/6$, these forever-honest nodes output 0 in $W_{h,0}$ and 1 in $W_{h,1}$. Lastly, we will show that with a constant probability, an honest node S cannot distinguish between $W_{h,0}$ and $W_{h,1}$, leading to a consistency violation with probability $> 1 - p$ in one of the two worlds. Note that the designated sender may be corrupted in $W_{h,b}$, so we need to show a violation of consistency, not validity.

World $W_{c,b}$: In $W_{c,b}$, node S is (statically) corrupt. All other nodes (including the designated sender) are honest and execute the protocol as specified. The corrupt node S simulates an execution in the $W_{c,1-b}$ world in its head for up to C multicasts. To elaborate, for every round, in addition to receiving messages from honest nodes in $W_{c,b}$, the corrupt node S simulates the receipt of messages multicast by all other nodes in world $W_{c,1-b}$, until C multicasts have occurred in the simulated execution. The corrupt node S treats the received messages (from both the real world $W_{c,b}$ and the simulated world $W_{c,1-b}$) as if they are from the same execution. It then sends multicast messages as instructed by an honest execution of the protocol. When node S multicasts a message in the real execution, its messages arrive in both the real as well as the simulated execution. We note that a simulation of $W_{c,1-b}$ by node S is possible only due to the non-existence of a trusted setup. In this scenario, messages sent by a node does not depend on any secret that is not known to S. On the other hand, if there was a trusted setup such as a PKI, then these messages could not have been simulated.

Observe that in world $W_{c,b}$, only node S is corrupted and the designated sender is honest. Hence, by the validity guarantee of the Byzantine broadcast protocol, we have: With probability $p > 5/6$, honest nodes in $W_{c,b}$ output b.

World $W_{h,b}$: In $W_{h,b}$, all nodes are honest at the start of the protocol and the adversary makes adaptive corruptions along the way. The adversary simulates a protocol execution in world $W_{h,1-b}$ in its head. Specifically, at the start of each round, the adversary simulates this round for all nodes *except node* S in $W_{h,1-b}$ in its head, and checks to see which nodes will send a message in this round of the simulated execution. Whenever a node j in the simulated execution wants to speak, if there have not been C multicast messages from nodes other than node S in this execution, the adversary adaptively corrupts node j (unless it is already corrupt) in world $W_{h,b}$. If node S multicasts messages in the real execution, its messages arrive in both the real as well as the simulated execution.

In a round, a corrupt node j does the following. It sends all messages as instructed for node j by the protocol. In addition, it sends the messages node j in $W_{h,1-b}$ would have sent to node S in this round; note that these messages are sent to node S only and not to anyone else.

Indistinguishability between worlds $W_{h,b}$ and $W_{c,b}$ for forever-honest nodes. The corrupt node S in $W_{c,b}$ behaves exactly like the honest node S in $W_{h,b}$. Corrupt nodes in $W_{h,b}$ behave honestly towards forever-honest nodes other than node S. Therefore, the views of the nodes that are forever-honest in both $W_{h,b}$ and $W_{c,b}$ are identically distributed. Let Y denote the event that these forever-honest nodes output b in $W_{h,b}$. Based on the indistinguishability and the aforementioned validity guarantee in $W_{c,b}$, we have $\Pr[Y] \geq p > 5/6$.

Indistinguishability between worlds $W_{h,b}$ and $W_{h,1-b}$ for node S. Observe that in both worlds, the honest node S receives all the messages in that world (through the honest protocol execution) and messages from the

first up to C nodes in the other world (through messages sent by adaptively corrupted nodes that would be sending honest messages in the simulated world). Thus, given that the honest and the simulated execution both have C multicast complexity, the view of node S in worlds $W_{h,b}$ and $W_{h,1-b}$ is identically distributed.

More formally, let A_r and A_s denote the events that the real and simulated executions respectively have C multicast complexity. Recall that the protocol satisfies consistency, validity and termination, and has C multicast complexity with probability p. Thus, $\Pr[A_r] \geq p$, $\Pr[A_s] \geq p$, and $\Pr[A_r \cap A_s] \geq \Pr[A_r] + \Pr[A_s] - 1 \geq 2p - 1$.

Let X denote the event that node S does not output 1. Given that the view of node S is identically distributed when the honest and the simulated executions both have C multicast complexity, without loss of generality, we have $\Pr[X|A_r \cap A_s] \geq 1/2$, and

$$\begin{aligned}\Pr[X] &\geq \Pr[X \cap A_r \cap A_s] = \Pr[X|A_r \cap A_s] \cdot \Pr[A_r \cap A_s] \\ &\geq \frac{1}{2}(2p-1) > 1/3.\end{aligned}$$

Consistency violation in $W_{h,1}$. The probability that consistency of Byzantine broadcast is violated is given by

$$\Pr[\text{consistency violation}] \geq \Pr[X \cap Y] > 1/3 + 5/6 - 1 = 1/6.$$

This contradicts the supposition that the protocol solves Byzantine broadcast with $> 5/6$ probability.

Remark. An interesting open question is show whether this bound is tight, i.e., whether there exists a protocol that can achieve C multicast complexity with C adaptive corruptions with probability $5/6$.

8 Conclusion

In this paper, we present three key results. We first show that any (randomized) BA protocol tolerating a strongly adaptive adversary capable of performing after-the-fact removal incurs a communication complexity of at least $\Omega(f^2)$ bits. We then show two protocols, one under synchrony tolerating up to one-half Byzantine faults and one under partial synchrony tolerating up to one-third Byzantine faults, with sub-quadratic communication complexity against an adaptive adversary that does not perform after-the-fact removal. Our protocols assume a PKI, but compared to prior work, it does not rely on memory erasures. Finally, we show the need for setup assumptions. In particular, we show that setup assumptions are necessary if a BA protocol can tolerate C adaptive corruptions and have a multicast complexity of C.

References

1. Abraham, I., Devadas, S., Dolev, D., Nayak, K., Ren, L.: Synchronous byzantine agreement with expected $O(1)$ rounds, expected $O(n^2)$ communication, and optimal resilience. In: Financial Crypto (2019)
2. Attiya, H., Welch, J.: Distributed Computing: Fundamentals, Simulations and Advanced Topics. John Wiley & Sons, Inc., USA (2004)
3. Ben-Or, M.: Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In: PODC (1983)
4. Bhangale, A., Liu-Zhang, C.D., Loss, J., Nayak, K.: Efficient adaptively-secure byzantine agreement for long messages. Cryptology ePrint Archive, Report 2021/1403 (2021). https://ia.cr/2021/1403
5. Bitansky, N.: Verifiable random functions from non-interactive witness-indistinguishable proofs. In: Theory of Cryptography, pp. 567–594 (2017)
6. Blum, E., Katz, J., Liu-Zhang, C., Loss, J.: Asynchronous byzantine agreement with subquadratic communication. In: R. Pass, K. Pietrzak (eds.) Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part I, *Lecture Notes in Computer Science*, vol. 12550, pp. 353–380. Springer (2020). DOI 10.1007/978-3-030-64375-1_13. URL https://doi.org/10.1007/978-3-030-64375-1_13
7. Cachin, C., Kursawe, K., Petzold, F., Shoup, V.: Secure and efficient asynchronous broadcast protocols. In: Annual International Cryptology Conference, pp. 524–541. Springer (2001)
8. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS (2001)
9. Castro, M., Liskov, B.: Practical byzantine fault tolerance. In: OSDI (1999)
10. Chandran, N., Chongchitmate, W., Garay, J.A., Goldwasser, S., Ostrovsky, R., Zikas, V.: The hidden graph model: Communication locality and optimal resiliency with adaptive faults. In: ITCS (2015)
11. Chen, J., Gorbunov, S., Micali, S., Vlachos, G.: Algorand agreement: Super fast and partition resilient byzantine agreement. Cryptology ePrint Archive, Report 2018/377 (2018). https://ia.cr/2018/377
12. Chen, J., Micali, S.: Algorand: The efficient and democratic ledger. https://arxiv.org/abs/1607.01341 (2016)
13. Cohen, R., Coretti, S., Garay, J., Zikas, V.: Probabilistic termination and composability of cryptographic protocols. In: the 36th Annual International Cryptology Conference on Advances in Cryptology — CRYPTO 2016, pp. 240–269. Springer (2016)
14. Cohen, S., Keidar, I., Spiegelman, A.: Brief announcement: Not a coincidence: Subquadratic asynchronous byzantine agreement whp. In: Proceedings of the 39th Symposium on Principles of Distributed Computing, PODC '20, p. 175177. Association for Computing Machinery, New York, NY, USA (2020). DOI 10.1145/3382734.3405708. URL https://doi.org/10.1145/3382734.3405708
15. David, B.M., Gazi, P., Kiayias, A., Russell, A.: Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In: Eurocrypt (2018)
16. Dolev, D., Reischuk, R.: Bounds on information exchange for byzantine agreement. J. ACM **32**(1), 191–204 (1985)
17. Dolev, D., Strong, H.R.: Authenticated algorithms for byzantine agreement. Siam Journal on Computing - SIAMCOMP **12**(4), 656–666 (1983)
18. Dryja, T., Liu, Q.C., Narula, N.: A lower bound for byzantine agreement and consensus for adaptive adversaries using vdfs. CoRR **abs/2004.01939** (2020). URL https://arxiv.org/abs/2004.01939
19. Dwork, C., Lynch, N., Stockmeyer, L.: Consensus in the presence of partial synchrony. J. ACM (1988)
20. Feldman, P., Micali, S.: Optimal algorithms for byzantine agreement. In: Proceedings of the twentieth annual ACM symposium on Theory of computing, pp. 148–161. ACM (1988)
21. Fischer, M.J., Lynch, N.A., Merritt, M.: Easy impossibility proofs for distributed consensus problems. In: PODC (1985)

22. Fitzi, M.: Generalized communication and security models in byzantine agreement. Ph.D. thesis, ETH Zurich (2002)
23. Garay, J.A., Katz, J., Kumaresan, R., Zhou, H.S.: Adaptively secure broadcast, revisited. In: Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, PODC '11, pp. 179–186. ACM, New York, NY, USA (2011)
24. Garay, J.A., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: Analysis and applications. In: Eurocrypt (2015)
25. Goyal, R., Hohenberger, S., Koppula, V., Waters, B.: A generic approach to constructing and proving verifiable random functions. In: TCC, vol. 10678, pp. 537–566. Springer (2017)
26. Groth, J., Ostrovsky, R., Sahai, A.: New techniques for noninteractive zero-knowledge. Journal of the ACM (JACM) **59**(3), 1–35 (2012)
27. Groth, J., Ostrovsky, R., Sahai, A.: New techniques for noninteractive zero-knowledge. J. ACM **59**(3), 11:1–11:35 (2012). DOI 10.1145/2220357.2220358. URL `http://doi.acm.org/10.1145/2220357.2220358`
28. Hirt, M., Zikas, V.: Adaptively secure broadcast. In: EUROCRYPT, vol. 6110, pp. 466–485. Springer (2010)
29. Katz, J., Koo, C.Y.: On expected constant-round protocols for byzantine agreement. J. Comput. Syst. Sci. **75**(2), 91–112 (2009)
30. King, V., Saia, J.: Breaking the $O(N^2)$ bit barrier: Scalable byzantine agreement with an adaptive adversary. J. ACM **58**(4), 18:1–18:24 (2011)
31. King, V., Saia, J., Sanwalani, V., Vee, E.: Scalable leader election. In: SODA (2006)
32. Lamport, L.: The weak byzantine generals problem. J. ACM **30**(3), 668–676 (1983)
33. Lamport, L., Shostak, R., Pease, M.: The byzantine generals problem. ACM Trans. Program. Lang. Syst. **4**(3), 382–401 (1982)
34. Micali, S., Vadhan, S., Rabin, M.: Verifiable random functions. In: FOCS (1999)
35. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
36. Rabin, M.O.: Randomized byzantine generals. In: Proceedings of the 24th Annual Symposium on Foundations of Computer Science, pp. 403–409. IEEE (1983)
37. Tsimos, G., Loss, J., Papamanthou, C.: Gossiping for communication-efficient broadcast. Cryptology ePrint Archive, Report 2020/894 (2020). `https://ia.cr/2020/894`

9 Additional Details on Modeling

9.1 Protocol Execution

In the main body, we omitted some details of the execution model for ease of understanding. We now explain these details that will be relevant to the formal proofs.

An external party called the environment and denoted $\mathcal{Z}$ provides inputs to honest nodes and receives outputs from the honest nodes. As mentioned, the 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, we assume that all players as well as $\mathcal{A}$ and $\mathcal{Z}$ are non-uniform, probabilistic polynomial-time (p.p.t.) Interactive Turing Machines, and the execution is parametrized by a security parameter κ that is common knowledge to all players as well as $\mathcal{A}$ and $\mathcal{Z}$. Since the 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.

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

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

- On receive $\texttt{mine}(\mathsf{m})$ from node i for the first time: let $\mathsf{Coin}[\mathsf{m}, i] := \mathsf{Bernoulli}(\mathfrak{p}(\mathsf{m}))$ 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.

Fig. 2 The mining ideal functionality $\mathcal{F}_{\text{mine}}$.

Notational conventions. In our subsequent proofs we sometimes use the notation view to denote a randomly sampled execution. The randomness in the execution 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.

9.2 Ideal Mining Functionality $\mathcal{F}_{\text{mine}}$

Earlier we used $\mathcal{F}_{\text{mine}}$ to describe our ideal-world protocols. For preciseness we now spell out the details of $\mathcal{F}_{\text{mine}}$.

$\mathcal{F}_{\text{mine}}$ ideal functionality. As shown in Figure 2, 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 with appropriate probability to decide if node i is eligible to send 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 is eligible to send m.

Recall in our scheme, different types of messages are associated with different probabilities, and we assume that this is hard-wired in $\mathcal{F}_{\text{mine}}$ with the mapping $\mathfrak{p}$ that maps each message type to an appropriate probability (see Figure 2). This $\mathcal{F}_{\text{mine}}$ functionality is secret since if a so-far-honest node i has not attempted to determine if it is eligible to send m, then no corrupt node can learn whether i is in the committee corresponding to m.

10 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 [8] — instead, we describe a real-world protocol that preserves all the security properties of the $\mathcal{F}_{\text{mine}}$-hybrid protocols.

10.1 Preliminary: Adaptively Secure Non-Interactive Zero-Knowledge Proofs

We use the same definitions for NIZK as in Groth et al. [27]. In particular, since we need adaptive security, the underlying NIZK must be adaptively secure as well. In the definitions below, the notion of "non-erasure computational zero-knowledge" captures the adaptive nature of the adversary, and the notion "perfect knowledge extraction" implies soundness.

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.

- $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\kappa)$: 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

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

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 exist 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[2] and without a witness, and then runs $r' \leftarrow \mathsf{Explain}(\mathsf{crs}_0, \tau_0, \mathsf{stmt}, w, \rho)$ and outputs (π, r').

In the above, Gen_0 is often called a simulated setup algorithm. The term τ_0 is often called a trapdoor which is need to generate simulated proofs without the witness, and crs_0 is often referred to as the simulated CRS.

[2] Here, we write the random coins ρ consumed by the algorithm P_0 explicitly, although in other places, without risk of ambiguity, we avoid writing the randomness explicitly consumed by randomized algorithms.

Perfect knowledge extraction. 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,

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

We often call Gen_1 a simulated setup, which generates a simulated common reference string denoted crs_1, and an extraction trapdoor τ_1. the extraction trapdoor is needed to extract a witness from a proof submitted by the adversary. The notion of perfect knowledge extraction implies standard soundness. We will need the slightly stronger knowledge extraction notion, therefore, we define it this way.

10.2 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 exist 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 $\mathsf{com}(\mathsf{crs}, v, \rho) = \mathsf{com}(\mathsf{crs}, v', \rho')$.

The existence of such a NIZK scheme was shown by Groth et al. [27] 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.

Theorem 16 (Instantiation of our NIZK and commitment schemes [27]) *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 exists a commitment scheme that is perfectly binding and computationally hiding under selective opening.*

10.3 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 $\mathsf{PRF}_{\mathsf{sk}}(\mathsf{m}) = \rho$.

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

Our real-world protocol 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, 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 our previous $\mathcal{F}_{\mathrm{mine}}$-hybrid protocol described in Section 5. The high-level idea is to realize an adaptively secure VRF from adaptively secure PRFs and NIZKs:

- **Trusted PKI setup.** Upfront, a trusted party runs the CRS generation algorithms of the commitment and the NIZK scheme to obtain $\mathsf{crs}_{\mathrm{comm}}$ and $\mathsf{crs}_{\mathrm{nizk}}$. It then chooses a secret PRF key for every node, where the i-th node has key sk_i. It publishes $(\mathsf{crs}_{\mathrm{comm}}, \mathsf{crs}_{\mathrm{nizk}})$ 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 $\mathcal{F}_{\mathrm{mine}}$.`mine`.** Recall that in the ideal-world protocol a node i calls $\mathcal{F}_{\mathrm{mine}}.\mathtt{mine}(\mathsf{m})$ 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

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

  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 (Section 5), m can be of the form $(\mathtt{T}, r, b)$ where $\mathtt{T} \in \{\texttt{Propose}, \texttt{Vote}, \texttt{Commit}, \texttt{Status}\}$, r denotes an epoch number, and $b \in \{0, 1, \perp\}$; or of the form $(\texttt{Terminate}, b)$.
  - Compound messages, i.e., a concatenation of the above types of messages.

  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 D_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.

10.5 Main Theorems for Real-World Protocols

After applying the above compiler to our $\mathcal{F}_{\text{mine}}$-hybrid protocols described in Sections 5 and 6, we obtain our real-world protocol. In this section, we present our main theorem statements for these three settings. The proofs for these theorems can be derived by combining the proofs in Section 5 and 6 as well as those in the following section, i.e., Appendix 11, where we will show that the relevant security properties are preserved in the real world as long as the cryptographic building blocks are secure.

In theorem statement below, 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 17 (Sub-quadratic BA under Synchrony) *Let π_{sync} be the protocol obtained by applying the above compiler to the protocol in Section 5, and assume that the cryptographic building blocks employed are secure. Then, for any arbitrarily small positive constant ϵ, any $n \in \mathbb{N}$, π_{sync} satisfies consistency and validity, and tolerates $f < (\frac{1}{2} - \epsilon)n$ adaptive corruptions, except for $\mathsf{negl}(\kappa)$ probability. Further, π_{sync} achieves expected constant round and $\chi \cdot \mathsf{poly}\log(\kappa)$ multicast complexity. In the above, χ is a security parameter related to the hardness of the cryptographic building blocks; $\chi = \mathsf{poly}(\kappa)$ under standard cryptographic assumptions and $\chi = \mathsf{poly}\log(\kappa)$ if we assume sub-exponential security of the cryptographic primitives employed.*

Theorem 18 (Sub-quadratic BA under Partial Synchrony) *Let $\pi_{\text{partialsync}}$ be the protocol obtained by applying the above compiler to the protocol in Section 6, 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 and validity, and tolerates $f < (\frac{1}{3} - \epsilon)n$ adaptive corruptions, except for $\mathsf{negl}(\kappa)$ probability. Further, $\pi_{\text{partialsync}}$ achieves expected $\Delta \cdot \mathsf{poly}\log(\kappa)$ rounds and $\chi \cdot \mathsf{poly}\log(\kappa) \cdot \log \Delta$ multicast complexity. In the above, χ is a security parameter related to the hardness of the cryptographic building blocks; $\chi = \mathsf{poly}(\kappa)$ under standard cryptographic assumptions and $\chi = \mathsf{poly}\log(\kappa)$ if we assume sub-exponential security of the cryptographic primitives employed.*

Proof Proofs for the above two theorems can be obtained by combining the $\mathcal{F}_{\text{mine}}$-hybrid analysis in Section 5 or 6 with Appendix 11 where we show that the relevant security properties are preserved in by the real world protocol.

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

In this section, we prove that our real-world protocol is just as secure as the $\mathcal{F}_{\text{mine}}$-hybrid protocol. To do this, we do not cryptographically realize $\mathcal{F}_{\text{mine}}$. Instead, our proof in essence defines a sequence of hybrids that gradually modifies the $\mathcal{F}_{\text{mine}}$-hybrid protocol to the real-world protocol, andwe prove that each step of modification does not decrease the security of the protocol (except for negligible amounts). We stress that this is different from the more typical proof that argues the adversary's views in adjacent hybrids to be computationally indistinguishable.One tricky aspect of the proof stems from the fact that we consider *adaptive* corruption. Whenever the real-world adversary adaptively

corrupts someone, the player's PRF key is revealed to the adversary.For this reason, our proof requires a PRF that resists selective-opening attacks, i.e., honest players' PRFs shouldstill be secure even when the adversary can adaptively corrupt some players and learn their secret keys. In Section 11.1, we define this selective opening notion of PRF security, and prove that we can get this notionfrom the standard PRF security. The most technical part of the proof is described in Section 11.3, where we gradually replace the PRF outcomes with random values as in the $\mathcal{F}_{\text{mine}}$-hybrid world. To achieve this, we describe a hybrid sequence that essentially goes back in time, where the k-th hybrid step replaces the honest users' PRF outcomes after the k-th corruption query with random coins. To make this hybrid sequence go through, we define some polynomial-time checkable bad events in Section 11.2.

11.1 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, every challenge tuple (i^*, m) it submits 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 1 (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 19 *Any secure PRF family satisfies pseudorandomness under selective opening by Definition 1 (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 1) 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).

11.2 Definition of Polynomial-Time Checkable Stochastic Bad Events

In all of our $\mathcal{F}_{\text{mine}}$-hybrid protocols earlier, some stochastic bad events related to $\mathcal{F}_{\text{mine}}$'s random coins can lead to the breach of protocol security (i.e., consistency, validity, or termination) 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.[3] More formally, for the honest majority protocol, the stochastic bad events are stated in Lemmas 2, 3, 4, and 6.

For these stochastic bad events, 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. Recall that view denotes an execution trace.

In our earlier $\mathcal{F}_{\text{mine}}$-world analyses (in Section 5), 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* — this is because the $\mathcal{F}_{\text{mine}}$-world protocol does not use any cryptography such as VRF, but abstracts the cryptography with an ideal functionality $\mathcal{F}_{\text{mine}}$.

[3] Note that too many honest mining successes will not break the security properties of the protocol, and thus we do not consider this as a bad event.

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.

Below we will start from the $\mathcal{F}_{\text{mine}}$ hybrid protocol and through a sequence of hybrids, arrive at the real-world protocol that uses actual cryptography. The same proof works for both the synchronous protocol and partially synchronous protocol.

11.3 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 8 *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 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 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 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 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 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 that 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 used in the claim in Hybrid $1.f$, 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 8.

11.4 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 9 *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 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 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 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.

11.5 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 10 *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 (see the definition of Perfect Knowledge Extraction earlier in Section 10.1). 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 11 *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 preserved 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.

11.6 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 12 *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 forgery event then either the relevant bad events occur both in Hybrid 3 and the real-world execution, or occur in neither. For views with forgery events, it is not difficult to see that if Hybrid 3 (on this view) does not incur the relevant bad events, then neither would the real-world execution (for this view).