Communication Complexity of Byzantine Agreement, Revisited

Ittai Abraham
VMware Research

T-H. Hubert Chan
The University of Hong Kong

Danny Dolev
The Hebrew University of Jerusalem

Kartik Nayak
VMware Research

Rafael Pass
Cornell Tech

Ling Ren
VMware Research

Elaine Shi
Cornell University

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.

Next, we show a new subquadratic binary BA construction (of course, assuming no after-the-fact removal) that achieves near-optimal resilience and expected constant rounds under standard cryptographic assumptions and a public-key infrastructure (PKI). 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.

ACM Reference Format:
Ittai Abraham, T-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. 2019. Communication Complexity of Byzantine Agreement, Revisited. In *2019 ACM Symposium on Principles of Distributed Computing (PODC '19), July 29-August 2, 2019, Toronto, ON, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3293611.3331629

1 INTRODUCTION

Byzantine agreement (BA) [24] is a central abstraction in distributed systems. Typical BA protocols [5, 11, 12] 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 [26]. 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 nodes 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 κ-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 [6, 22]. Once a committee is selected, we can run any BA protocol among the committee, and let the committee members may 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 $\exp(-\Omega(\kappa))$ 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 what nodes 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 [21] 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 [17, 26] can reach agreement in subquadratic communication assuming idealized proof-of-work. Subsequently, several so-called "proof-of-stake" constructions [7, 9] also showed how to realize BA with subquadratic communication. All of the above works 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 node 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, 11, 20] 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 node i sends a message m in round r, a strongly adaptive adversary (e.g., who

PODC '19, July 29-August 2, 2019, Toronto, ON, Canada

ACM ISBN 978-1-4503-6217-7/19/07…$15.00
https://doi.org/10.1145/3293611.3331629

controls the egress routers of many nodes) can observe m and then decide to corrupt i and erase the message m that node 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 $\Omega(f^2)$ communication in expectation in the presence of a strongly adaptive adversary where f denotes the number of corrupt nodes. The proof of our lower bound is inspired by the work of Dolev and Reischuk [10], who showed that any *deterministic* BA protocol must incur $\Omega(f^2)$ communication 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 nodes, 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 nodes.*

Main result 2: near-optimal subquadratic BA with minimal assumptions. On the upper bound front, we present a subquadratic BA protocol that, besides the necessary "no after-the-fact removal" assumption, relies only on standard cryptographic and setup assumptions. Our protocol achieves 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 [7, 9] or proof-of-work [26]. In particular, some works [7, 21] assume the ability of honest nodes to securely erase secrets from memory and that adaptive corruption cannot take place between when an honest node 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 [7]. 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 node to multicast the same message to everyone, than to unicast n different messages (of the same length) to n different nodes — even though the two have identical communication complexity in the standard pair-wise 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 messages 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), for any constant $0 < \epsilon < 1/2$, there exists a synchronous BA protocol with expected $O(\kappa)$ multicast complexity, expected $O(1)$ round complexity, and $\exp(-\Omega(\kappa))$ error probability that tolerates $f < (1-\epsilon)n/2$ adaptively corrupted players out of n players in total.*

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 nodes' secret keys, which is a different notion of adaptivity from in some prior works [4, 18]. Most previously known VRF constructions [4, 18, 25] do not provide security under an adaptive adversary. Chen and Micali [7] use random oracles (RO) and unique signatures to construct an adaptively secure VRF. In the full version of this paper, we will show how to instantiate an adaptively secure VRF from standard cryptographic assumptions such as bilinear groups.

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 nodes (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 nodes to speak [21].

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. Section 3 proves Theorem 1. Section 4 and 5 construct an adaptively secure BA protocol to prove Theorem 2. Section 6 proves Theorem 3.

1.1 Related Work

Dolev and Reischuk [10] 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, 27] to achieve expected constant round complexity [1, 13, 20] even under a strongly adaptive adversary. A line of works [8, 16, 19] 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 [21]. More recently Nakamoto-style protocols, based on either proof-of-work [26] or proof-of-stake [7, 9] also showed how to realize BA with subquadratic communication. Compared to our protocol in Section 5, these existing works make other strong assumptions, and even with those strong assumptions, cannot simultaneously achieve near-optimal resilience and expected constant rounds. Nakamoto consensus [26] assumes idealized proofs-of-work. Proof-of-stake protocols assume random oracles [7, 9]. King-Saia [21] and Chen-Micali [7] assume memory-erasure. Nakamoto-style protocols [9, 26] and King-Saia [21] cannot achieve expected constant rounds. Chen-Micali [7] have sub-optimal tolerance of $f < (\frac{1}{3} - \epsilon)n$.

2 MODEL AND DEFINITION

Communication model. We assume that 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. We measure communication complexity by the number of messages sent by honest nodes. Our protocols in Section 4 and 5 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 multicasts by honest nodes is bounded by C.

Adversary model. Prior to the protocol execution, each node generates its public/private key pair honestly and sends its public key to all other nodes. The adversary is denoted $\mathcal{A}$. $\mathcal{A}$ can *adaptively* corrupt nodes any time during the protocol execution after the trusted setup. The total number of corrupt nodes at the end of the execution is at most f. 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 protocol are referred to as *forever-honest* nodes. 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. However, the message m that was already multicast before i became corrupt *cannot be retracted* — it will be received by all so-far-honest nodes at the beginning of the next round. In other words, the adversary is adaptive but not strongly adaptive (i.e., incapable of after-the-fact removal).

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 [15] while the broadcast version can tolerate up to $n - 1$ corruptions [11, 24]. 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 [10, 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. 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 parties 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 party in a given round r, decide to adaptively corrupt h, and then remove 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, then in expectation, honest nodes collectively need to send at least $(\epsilon f)^2$ 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 $(\epsilon f)^2$ 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 larger than the above.

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, then 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 $f/2$ 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 $f/2 < \lceil n/2 \rceil$). 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 $f/2$ messages sent to it by nodes in U.

For a protocol to have an expected message complexity of $(\epsilon f)^2$, 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 $E[z] < (\epsilon f)^2$. Let X_1 be the event that $z \le \frac{\epsilon}{2} f^2$. By Markov's inequality, $\Pr[z > \frac{1}{2\epsilon} E[z]] < 2\epsilon$. Thus, $\Pr[z \le \frac{\epsilon}{2} f^2] \ge \Pr[z \le \frac{1}{2\epsilon} E[z]] > 1 - 2\epsilon$.

Let X_2 be the event that among the first $\frac{\epsilon}{2} f^2$ messages, a node p picked uniformly at random from V by the adversary receives at most $f/2$ messages. Observe that among the first $\frac{\epsilon}{2} f^2 = 2\epsilon |V| (f/2)$ messages, there exist at most $2\epsilon|V|$ nodes that receive more than $f/2$ of those. Since p has been picked uniformly at random from V, $\Pr[X_2] \ge 1 - 2\epsilon$. 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 identically as $\mathcal{A}$ 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 $f/2$ 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 $f/2$ 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 $f/2$ messages from U to p have been blocked, corrupted 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 $\frac{\epsilon}{2} f^2$ and among those, the randomly picked node p has received at most $f/2$ 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 $\mathcal{A}'$. Recall that Y_1 includes the event that p outputs 1 as well as the event that p does not terminate. We have $\Pr[Y_1] \ge \Pr[Y_1 | X_1 \cap X_2] \cdot \Pr[X_1 \cap X_2] > \frac{1}{2}(1 - 4\epsilon)$.

Meanwhile, we argue that honest nodes in U cannot distinguish $\mathcal{A}$ and $\mathcal{A}'$. This is because the only difference between the two scenarios is that, under $\mathcal{A}$, the first $f/2$ messages from U to p are intentionally ignored by a corrupt node p, and under $\mathcal{A}'$, the first $f/2$ messages from U to an honest p are blocked by $\mathcal{A}'$ using after-the-fact removal. Thus, honest nodes in U receive identical messages under $\mathcal{A}$ and $\mathcal{A}'$ and cannot distinguish the two adversaries. Under $\mathcal{A}$, they need to output 0 to preserve validity. Recall that the protocol solves Byzantine broadcast with at least $\frac{3}{4} + \epsilon$ probability. Thus, with at least the above 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 they cannot distinguish $\mathcal{A}$ and $\mathcal{A}'$, $\Pr[Y_2] \ge \frac{3}{4} + \epsilon$.

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

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

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

4 SUBQUADRATIC BA: $f < (1/3 - \epsilon)n$

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 $O(\kappa)$ 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 κ iterations $r = 1, 2, \ldots \kappa$, 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 works because of the following observations. 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. Except with $\exp(-\Omega(\kappa))$ 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) [25]. 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.

Strawman: the Chen-Micali approach. We first describe the paradigm of Chen and Micali [7] 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 $\mathsf{VRF}^1_{\mathsf{sk}_i}(\texttt{Vote}, r) < D$ to determine whether i is eligible to vote in round r where D is a difficulty parameter appropriately chosen such that, in expectation, κ 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 $\frac{2\kappa}{3}$, 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 player 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 [7] 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 difficult 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 κ-sized in expectation; whereas
(2) D_0 should be set such that every node has a $\frac{1}{2n}$ probability to be eligible to propose.

Since we are interested in making communication scale better with n, we assume $n = \omega(\kappa)$ and $n > \kappa$; otherwise, one should simply use the quadratic protocol.

Putting it together. More formally, we use the phrase "node i *conditionally multicasts* a message (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 $(\mathsf{T}, r, b, i, \pi)$, where $\mathsf{T} \in \{\texttt{Propose}, \texttt{Vote}\}$ stands for the type of the message and π is a proof proving that i indeed is eligible (note that π includes both the pseudorandom evaluation result and the proof output by the VRF). 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 $\frac{2\kappa}{3}$; 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 (T, r, b) where $\mathsf{T} \in \{\texttt{Propose}, \texttt{Vote}\}$, it calls $\mathcal{F}_{\text{mine}}.\texttt{mine}(\mathsf{T}, r, b)$, and $\mathcal{F}_{\text{mine}}$ shall flip a random coin with appropriate probability to determine whether this "mining" attempt is successful. If successful, $\mathcal{F}_{\text{mine}}.\texttt{verify}((\mathsf{T}, r, b), i)$ 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 (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 $\frac{\kappa}{n}$. Hence, if there are $\frac{2\kappa}{3}$ votes for each of the bits 0 and 1, this means there are at least in total $\frac{4\kappa}{3}$ successful mining attempts, which happens with $\exp(-\Omega(\kappa))$ probability, by the Chernoff bound. Therefore, except with $\exp(-\Omega(\kappa))$ probability, if any node sees $\frac{2\kappa}{3}$ votes for some bit b, then no other node sees $\frac{2\kappa}{3}$ 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 $\frac{1}{2n}$ 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 κ iterations, a good iteration exists except with $\exp(-\Omega(\kappa))$ 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 (T, r, b), and immediately corrupt the node in the same round and make it send $(\mathsf{T}, r, 1-b)$ too. If T is `Vote`, whenever there are $\frac{2\kappa}{3}$ votes for b in iteration r, by corrupting all these nodes that are eligible to vote, the adversary can construct $\frac{2\kappa}{3}$ 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, by corrupting this leader, the adversary gets a corrupt leader, and thus no good iteration would exist.

5 SUBQUADRATIC BA: $f < (1/2 - \epsilon)n$

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. 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. 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 $(\mathtt{Status}, r, b, C)$ containing the highest certified bit b it has seen so far as well as the corresponding certificate C.

(2) `Propose`. The leader L_r chooses a bit b with a highest certificate denoted C breaking ties arbitrarily. The leader multicasts $(\mathtt{Propose}, r, b)$. 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 $(\mathtt{Vote}, r = 1, b)$.
For all iterations $r \geq 2$, if a validly signed $(\mathtt{Propose}, r, b)$ message has been received from L_r with a certificate 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 $(\mathtt{Vote}, r, b)$ 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 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 C for b) and no iteration-r vote for $1-b$, it multicasts an iteration-r `Commit` message for b of the form $(\mathtt{Commit}, r, b)$ with the certificate 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 $(\mathtt{Terminate}, b)$ with the $f + 1$ `Commit` messages attached. The node then outputs b and terminates. This last message will make all other nodes multicast `Terminate`, 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 i 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 very 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 a (T, r, b)

where T is Status, Vote, or Commit, iff

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

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

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

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 each mining attempt for Status/Vote/Commit/Terminate has a κ/n probability to be successful and each mining attempt for leader proposal has a $1/2n$ probability to be successful. As before, we assume $n = \omega(\kappa)$ and $n > \kappa$; 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 $\kappa/2$ messages of that type; 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 $(\texttt{T}, r, b)$ where $\texttt{T} \in \{\texttt{Status}, \texttt{Propose}, \texttt{Vote}, \texttt{Commit}\}$ 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.

To prove consistency and validity, we first establish two lemmas, Lemma 5 and 6, to show that each bad event we care about happens with $\exp(-\Omega(\kappa))$ probability. We will then show there are at most $\text{poly}(\kappa)$ such bad events that we need to take a union bound over, so the overall error probability is still exponentially small in κ. Recall that $n > \kappa$ and that the adversary can make at most $(1/2 - \epsilon)n$ adaptive corruptions where $0 < \epsilon < 1/2$ is a constant.

Lemma 5. *Except for* $\exp(-\Omega(\kappa))$ *probability, if* $\epsilon n/2$ *honest nodes have terminated, all honest nodes terminate in the next round.*

Proof. Each of those $\epsilon n/2$ nodes has a κ/n probability to be eligible to send Terminate. The probability that none of them is eligible is $(1 - \kappa/n)^{\epsilon n/2} < \exp(-\epsilon\kappa/2) = \exp(-\Omega(\kappa))$. 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, a Terminate message sent by an honest eligible node makes all honest nodes terminate in the next round. □

Lemma 6. *Except for* $\exp(-\Omega(\kappa))$ *probability, For any* Status/Vote/Commit *message for bit* b *in iteration* r*, (i) less than* $\kappa/2$ *already-corrupt nodes are eligible to send it, and (ii) either* $\epsilon n/2$ *honest nodes have terminated, or at least* $\kappa/2$ *so-far-honest not-yet-terminated nodes are eligible to send it.*

Proof. For (i), observe that there are at most $(\frac{1}{2} - \epsilon)n$ already-corrupt nodes at any time. By our choice of D, in expectation, at most $(\frac{1}{2} - \epsilon)\kappa$ already-corrupt nodes are eligible to send the said message. A simple Chernoff bound completes the proof.

For (ii), if the "either" part is not true, then there are at least $(\frac{1}{2} + \frac{\epsilon}{2})n$ so-far-honest nodes that have not terminated. By our choice of D, in expectation, at least $(\frac{1}{2} + \frac{\epsilon}{2})\kappa$ of them are eligible to send the said message. A simple Chernoff bound completes the proof. □

Theorem 7 (Consistency). *Except for* $\exp(-\Omega(\kappa))$ *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 $\kappa/2$ Commit messages for b. By Lemma 6, except for $\exp(-\Omega(\kappa))$ probability, one of the Commit messages was sent by a so-far-honest node henceforth indexed by i^*. Similarly, for an iteration-r certificate for $1 - b$ to exist, except for $\exp(-\Omega(\kappa))$ probability, a so-far-honest node has multicast a vote for $1 - b$. But in that case, i^* would have received this conflicting vote and thus, still being honest by then, 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 $\exp(-\Omega(\kappa))$ probability.

Furthermore, by the beginning of iteration $r + 1$, all so-far 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 honest node will vote for $1 - b$ in iteration $r + 1$. To form a certificate for $1 - b$ in a subsequent iteration, all $\kappa/2$ votes have to come from already-corrupt nodes, which happens with $\exp(-\Omega(\kappa))$ probability by Lemma 6. □

Theorem 8 (Validity). *Except for* $\exp(-\Omega(\kappa))$ *probability, if all honest nodes have the same input bit* b*, then all nodes will output* b*.*

Proof. Straightforward from Lemma 6: in the first iteration, except for the said probability, there will be sufficient honest nodes to send $(\texttt{Vote}, r = 1, b)$, and there will not be sufficient corrupt nodes to vote for $1 - b$. Validity then follows from consistency. □

We then turn to analyze round complexity and 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 9. *Every iteration independently has a probability at least* $\frac{1}{2e}$ *to be a good iteration.*

Proof. In any fixed iteration r, there are $2n$ total attempts to propose (every node can attempt to propose 0 or 1). The probability that exactly one of these attempts succeeds is $\binom{2n}{1}\frac{1}{2n}(1 - \frac{1}{2n})^{2n-1}$. It is not hard to show (using derivatives) that the above expression decreases as n increases and is greater than $1/e$. With at least $1/2$

probability, this successful propose attempt comes from a so-far-honest node. Thus, every iteration independently has at least $\frac{1}{2e}$ probability to be a good iteration. □

Theorem 10 (Efficiency). *In expectation, all honest nodes terminate in $O(1)$ rounds and collectively send $O(n\kappa)$ messages (i.e., $O(\kappa)$ multicasts).*

Proof. By Lemma 9, with at least $\frac{1}{2e}$ probability, a single so-far-honest node is elected leader in an iteration. After this honest leader multicasts a unique proposal, all honest nodes will output and terminate in three rounds, unless there are insufficient eligible honest nodes to send `Vote` or `Commit` messages. Each of the above bad event happens with $\exp(-\Omega(\kappa))$ probability by Lemma 6. Thus, in each iteration, there is a $\frac{1}{2e} - \exp(-\Omega(\kappa)) = \Theta(1)$ probability that all nodes terminate. The expected constant round complexity thus follows in a straightforward fashion. In each round, in expectation, at most κ so-far-honest nodes multicast messages. The expected $O(n\kappa)$ communication complexity thus follows. □

Corollary 11 (Efficiency). *Except for $\exp(-\Omega(\kappa))$ probability, all honest nodes terminate in $O(\kappa)$ rounds and collectively send $O(n\kappa^2)$ messages (i.e., $O(\kappa^2)$ multicasts).*

Proof. The probability that none of consecutive κ iterations is good is $(1 - \frac{1}{2e})^\kappa = \exp(-\Omega(\kappa))$. □

Theorem 12. *For any constant $0 < \epsilon < 1/2$, the protocol in this section solves Byzantine agreement with $1 - \exp(-\Omega(\kappa))$ probability, terminates in expected $O(1)$ rounds, and achieves expected $O(\kappa)$ multicast complexity.*

Proof. Follows from Theorem 7, 8, and 10. □

6 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 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 13. *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.*

Although the lower bound is stated for multicast-based protocols, the same bound applies to a more general class of protocols in which at most C nodes send messages with $p > 5/6$ probability. In addition, the lower bound holds even when assuming the existence of a random oracle or a memory-erasure model.

Our proof is inspired by the classical techniques for proving consensus lower bounds in the authenticated channel model [14, 23, 24]; 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.

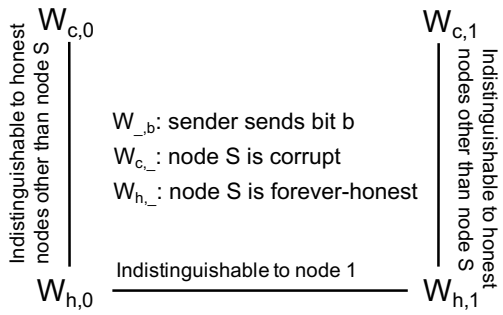

Figure 1: Relationships between different worlds in the sublinear multicast complexity without setup assumptions.

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, since the designated sender is honest in $W_{c,b}$, 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 $W_{h,1-b}$ by node S is possible only due to the non-existence of a trusted setup.

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. □

Acknowledgement. This work is partially supported by The Federmann Cyber Security Center in conjunction with the Israel National Cyber Directorate.

REFERENCES

[1] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. 2019. Synchronous byzantine agreement with expected $O(1)$ rounds, expected $O(n^2)$ communication, and optimal resilience. In *Financial Crypto*.
[2] Hagit Attiya and Jennifer Welch. 2004. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons, Inc., USA.
[3] Michael Ben-Or. 1983. Another Advantage of Free Choice (Extended Abstract): Completely Asynchronous Agreement Protocols. In *PODC*.
[4] Nir Bitansky. 2017. Verifiable Random Functions from Non-interactive Witness-Indistinguishable Proofs. In *Theory of Cryptography*. 567–594.
[5] Miguel Castro and Barbara Liskov. 1999. Practical Byzantine Fault Tolerance. In *OSDI*.
[6] Nishanth Chandran, Wutichai Chongchitmate, Juan A. Garay, Shafi Goldwasser, Rafail Ostrovsky, and Vassilis Zikas. 2015. The Hidden Graph Model: Communication Locality and Optimal Resiliency with Adaptive Faults. In *ITCS*.
[7] Jing Chen and Silvio Micali. 2016. ALGORAND: The Efficient and Democratic Ledger. https://arxiv.org/abs/1607.01341.
[8] Ran Cohen, Sandro Coretti, Juan Garay, and Vassilis Zikas. 2016. Probabilistic Termination and Composability of Cryptographic Protocols. In *the 36th Annual International Cryptology Conference on Advances in Cryptology — CRYPTO 2016*. Springer, 240–269.
[9] Bernardo Machado David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. 2018. Ouroboros Praos: An Adaptively-Secure, Semi-synchronous Proof-of-Stake Blockchain. In *Eurocrypt*.
[10] Danny Dolev and Rüdiger Reischuk. 1985. Bounds on Information Exchange for Byzantine Agreement. *J. ACM* 32, 1 (Jan. 1985), 191–204.
[11] Danny Dolev and H. Raymond Strong. 1983. Authenticated Algorithms for Byzantine Agreement. *Siam Journal on Computing - SIAMCOMP* 12, 4 (1983), 656–666.
[12] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. 1988. Consensus in the Presence of Partial Synchrony. *J. ACM* (1988).
[13] Paul Feldman and Silvio Micali. 1988. Optimal algorithms for Byzantine agreement. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*. ACM, 148–161.
[14] Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. 1985. Easy Impossibility Proofs for Distributed Consensus Problems. In *PODC*.
[15] Matthias Fitzi. 2002. *Generalized communication and security models in Byzantine agreement*. Ph.D. Dissertation. ETH Zurich.
[16] Juan A. Garay, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. 2011. Adaptively Secure Broadcast, Revisited. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC '11)*. ACM, New York, NY, USA, 179–186.
[17] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. 2015. The Bitcoin Backbone Protocol: Analysis and Applications. In *Eurocrypt*.
[18] Rishab Goyal, Susan Hohenberger, Venkata Koppula, and Brent Waters. 2017. A Generic Approach to Constructing and Proving Verifiable Random Functions. In *TCC*, Vol. 10678. Springer, 537–566.
[19] Martin Hirt and Vassilis Zikas. 2010. Adaptively Secure Broadcast. In *EUROCRYPT (Lecture Notes in Computer Science)*, Vol. 6110. Springer, 466–485.
[20] Jonathan Katz and Chiu-Yuen Koo. 2009. On Expected Constant-round Protocols for Byzantine Agreement. *J. Comput. Syst. Sci.* 75, 2 (Feb. 2009), 91–112.
[21] Valerie King and Jared Saia. 2011. Breaking the $O(N^2)$ Bit Barrier: Scalable Byzantine Agreement with an Adaptive Adversary. *J. ACM* 58, 4 (July 2011), 18:1–18:24.
[22] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. 2006. Scalable Leader Election. In *SODA*.
[23] Leslie Lamport. 1983. The Weak Byzantine Generals Problem. *J. ACM* 30, 3 (1983), 668–676.
[24] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.* 4, 3 (July 1982), 382–401.
[25] Silvio Micali, Salil Vadhan, and Michael Rabin. 1999. Verifiable Random Functions. In *FOCS*.
[26] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).
[27] Michael O. Rabin. 1983. Randomized Byzantine Generals. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science*. IEEE, 403–409.